

ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE ED  
 OBJECT MODULE PLACED IN ED.OBJ  
 COMPILER INVOKED BY: PLM80 ED.PLN OPTIMIZE DEBUG

```

1      $ TITLE(' CP/M-80 3.0 --- ED')
      ED;
      DO;
          /* MODIFIED FOR .PRL OPERATION MAY, 1979 */
          /* MODIFIED FOR OPERATION WITH CP/M 2.0 AUGUST 1979 */
          /* modified for MP/M 2.0 June 1981 */
          /* modified for CP/M 1.1 Oct 1981 */
          /* modified for CONCURRENT CP/M 1.0 Jul 1982 */
          /* modified for CP/M 3.0 July 1982 */
          /* modified for CP/M 3.0 SEPT 1982 */

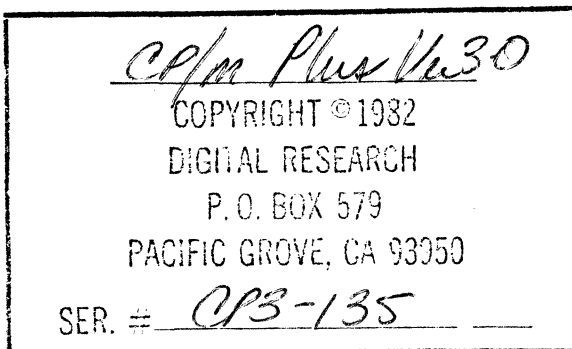
      /* MODIFICATION LOG:
      *   July 1982 whf: some code cleanup (grouped logicals, declared BOOL);
      *       fixed disk full error handling; fixed read from null files;
      *       fixed (some) of the dirty fcb handling (shouldn't use settype
      *       function on open fcbs!).
      *   July 1982 dh: installed patches to change macro abort command from
      *       ^C to ^Y and to not print error message when trying to delete
      *       a file that doesn't exist. Added PERROR: PROCEDURE to print
      *       error messages in a consistent format and modified error
      *       message handler at RESET: entry point. Also corrected Invalid
      *       filename error to not abort ED if parsing a R or X command.
      *       Modified start (at PLM:) and SETDEST: to prompt for missing
      *       filenames. Modified parsefcb & parse$lib to set a global
      *       flag and break if it got an invalid filename for X or R commands.
      *       Start sets page size from the system control block (SCB) if
      *       ED is running under CP/M-80 (high(ver)=0).
      *       The H command now works with new files. (sets newfile=false)
      *   Sept 82
      *       Corrected bug in which ED file b: didn't work. Changed PLM:
      *       and SETDEST: routines.
      *   Nov 82
      *       Corrected bug in parsefcb where filenames of 9 characters and
      *       types of 4 characters were accepted as valid and truncated.
      */

      $include (copyrt.lit)
      =
      = /*
      =   Copyright (C) 1982
      =   Digital Research
      =   P.O. Box 579
      =   Pacific Grove, CA 93950
      =   */
      =

2 1  declare
      mpmpduct literally '01h', /* requires mp/m */
      cp#3      literally '30h'; /* requires 3.0 cp/m */

3 1  declare plm label public; /* entry point for plm86 interface */

```



```
/* THE FOLLOWING COMMANDS CREATE ED.COM AND ED.CMD:
```

```

wm $1.plm
attach b 5
b:se eof $1.plm
vax $1.plm $$san\batch smpcmd $1 date($2 Oct 81)\
b:is14
ERA $1.MOD
era $1
era $1.obj
:f1:PLM80 $1.PLM debug PAGewidth(132) $3
:f1:link $101.obj,$1.obj,:f1:plm80.lib to $1.mod
:f1:locate $1.mod code(0100H) stacksize(100) map print($1,tra)
:f1:cpm
b:objcpm $1
attach b 1

```

the following VAX commands were used to create ED.CMD

```

$ asm86 scd1.a86 debug xref
! scd1 does a jump to the plm code
$ plm86 'p1'.plm 'p2' 'p3' 'p4' optimize(3) debug
$ link86 scd1.obj,'p1'.obj to 'p1'.lnk
$ loc86 'p1'.lnk od(sm(dats,code,data,stack,const)) ad(sm(code(0))) ss(stack(+16))
$ h86 'p1'

```

followed by the gencmd command  
gencmd ed data[b1E3,m80,xFFF]  
where 1E2 is the start of the constant area / 16 from ED.MP2

```
*/
```

```

/* DECLARE                                8080 Interface
JMP EDCOMMAND - 3 (TO ADDRESS LXI SP)
EDJMP BYTE DATA(0C3H),
EDADR ADDRESS DATA(.EDCOMMAND-3); */

```

```

/*****
*      B D O S   I N T E R F A C E
*
*****/

```

```

4 1  nonl:
      procedure (func,info) external;
5 2      declare func byte;
6 2      declare info address;
7 2      end nonl;

8 1  mon2:
      procedure (func,info) byte external;
9 2      declare func byte;
10 2     declare info address;

```

COPYRIGHT ©1982  
 DIGITAL RESEARCH  
 P. O. BOX 579  
 PACIFIC GROVE, CA 93950  
 SER. # \_\_\_\_\_

```
11 2      end mon2;

12 1      mon3;
          procedure (func,info) address external;
13 2          declare func byte;
14 2          declare info address;
15 2      end mon3;

16 1      declare fcb (1)  byte    external; /* 1st default fcb */
17 1      declare fcb16 (1) byte    external; /* 2nd default fcb */
18 1      declare tbuff (1) byte    external; /* default dma buffer */

19 1      DECLARE
          MAXB      ADDRESS EXTERNAL, /* MAX BASE 0006H */
          BUFF (128) BYTE    EXTERNAL, /* BUFFER 0080H */
          SECTSHF  LITERALLY '7', /* SHL(1,SECTSHF) = SECTSIZE */
          SECTSIZE LITERALLY '80H'; /* SECTOR SIZE */

20 1      BOOT: PROCEDURE ;
21 2          call mon1(0,0); /* changed for HP/M-86 version */
          /* SYSTEM REBOOT */
22 2      END BOOT;
```

COPYRIGHT ©1982  
DIGITAL RESEARCH  
P. O. BOX 579  
PACIFIC GROVE, CA 93950

SER. # \_\_\_\_\_

\$ eject

/\* ED : THE CP/M CONTEXT EDITOR \*/

/\* COPYRIGHT (C) 1976, 1977, 1978, 1979, 1980, 1981, 1982  
DIGITAL RESEARCH  
BOX 579 PACIFIC GROVE  
CALIFORNIA 93950

Revised:

07 April 81 by Thomas Rolander  
21 July 81 by Doug Huskey  
29 Oct 81 by Doug Huskey  
10 Nov 81 by Doug Huskey  
08 July 82 by Bill Fidler  
26 July 82 by Doug Huskey

\*/

/\* DECLARE COPYRIGHT(\*) BYTE DATA  
( ' COPYRIGHT (C) 1982, DIGITAL RESEARCH ' );  
\*\*\* this message should be in the header \*\*\*

\*/

23 1 declare date(\*) byte data ('8/82');

/\* COMMAND

FUNCTION

|             |                                                                                                       |
|-------------|-------------------------------------------------------------------------------------------------------|
| A           | APPEND LINES OF TEXT TO BUFFER                                                                        |
| B           | MOVE TO BEGINNING OR END OF TEXT                                                                      |
| C           | SKIP CHARACTERS                                                                                       |
| D           | DELETE CHARACTERS                                                                                     |
| E           | END OF EDIT                                                                                           |
| F           | FIND STRING IN CURRENT BUFFER                                                                         |
| H           | MOVE TO TOP OF FILE (HEAD)                                                                            |
| I           | INSERT CHARACTERS FROM KEYBOARD<br>UP TO NEXT <ENDFILE>                                               |
| J           | JUXTAPOSITION OPERATION - SEARCH FOR FIRST STRING,<br>INSERT SECOND STRING, DELETE UNTIL THIRD STRING |
| K           | DELETE LINES                                                                                          |
| L           | SKIP LINES                                                                                            |
| M           | MACRO DEFINITION (SEE COMMENT BELOW)                                                                  |
| N           | FIND NEXT OCCURRENCE OF STRING<br>WITH AUTO SCAN THROUGH FILE                                         |
| O           | RE-EDIT OLD FILE                                                                                      |
| P           | PAGE AND DISPLAY (MOVES UP OR DOWN 24 LINES AND<br>DISPLAYS 24 LINES)                                 |
| Q           | QUIT EDIT WITHOUT UPDATING THE FILE                                                                   |
| R<FILENAME> | READ FROM FILE <FILENAME> UNTIL <ENDFILE> AND<br>INSERT INTO TEXT                                     |
| S           | SEARCH FOR FIRST STRING, REPLACE BY SECOND STRING                                                     |
| T           | TYPE LINES                                                                                            |
| U           | TRANSLATE TO UPPER CASE (-U CHANGES TO NO TRANSLATE)                                                  |
| W           | WRITE LINES OF TEXT TO FILE                                                                           |
| X<FILENAME> | TRANSFER (XFER) LINES TO FILE <FILENAME>                                                              |
| Z           | SLEEP FOR 1/2 SECOND (USED IN MACROS TO STOP DISPLAY)                                                 |
| <CR>        | MOVE UP OR DOWN AND PRINT ONE LINE                                                                    |

COPYRIGHT ©1982  
DIGITAL RESEARCH  
P.O. BOX 579  
PACIFIC GROVE, CA 93950

SER. #

IN GENERAL, THE EDITOR ACCEPTS SINGLE LETTER COMMANDS WITH OPTIONAL

INTEGER VALUES PRECEDING THE COMMAND. THE EDITOR ACCEPTS BOTH UPPER AND LOWER CASE COMMANDS AND VALUES, AND PERFORMS TRANSLATION TO UPPER CASE UNDER THE FOLLOWING CONDITIONS. IF THE COMMAND IS TYPED IN UPPER CASE, THEN THE DATA WHICH FOLLOWS IS TRANSLATED TO UPPER CASE. THUS, IF THE 'I' COMMAND IS TYPED IN UPPER CASE, THEN ALL INPUT IS AUTOMATICALLY TRANSLATED (ALTHOUGH ECHOED IN LOWER CASE, AS TYPED). IF THE 'A' COMMAND IS TYPED IN UPPER CASE, THEN ALL INPUT IS TRANSLATED AS READ FROM THE DISK. GLOBAL TRANSLATION TO UPPER CASE CAN BE CONTROLLED BY THE 'U' COMMAND (-U TO NEGATE ITS EFFECT). IF YOU ARE OPERATING WITH AN UPPER CASE ONLY TERMINAL, THEN OPERATION IS AUTOMATIC. SIMILARLY, IF YOU ARE OPERATING WITH A LOWER CASE TERMINAL, AND TRANSLATION TO UPPER CASE IS NOT SPECIFIED, THEN LOWER CASE CHARACTERS CAN BE ENTERED.

A NUMBER OF COMMANDS CAN BE PRECEDED BY A POSITIVE OR NEGATIVE INTEGER BETWEEN 0 AND 65535 (1 IS DEFAULT IF NO VALUE IS SPECIFIED). THIS VALUE DETERMINES THE NUMBER OF TIMES THE COMMAND IS APPLIED BEFORE RETURNING FOR ANOTHER COMMAND.

#### THE COMMANDS

C D K L T P U <CR>

CAN BE PRECEDED BY AN UNSIGNED, POSITIVE, OR NEGATIVE NUMBER, THE COMMANDS

A F J N W Z

CAN BE PRECEDED BY AN UNSIGNED OR POSITIVE NUMBER, THE COMMANDS

E H O Q

CANNOT BE PRECEDED BY A NUMBER. THE COMMANDS

F I J M R S

ARE ALL FOLLOWED BY ONE OR MORE STRINGS OF CHARACTERS WHICH CAN BE OPTIONALLY SEPARATED OR TERMINATED BY EITHER <ENDFILE> OR <CR>. THE <ENDFILE> IS GENERALLY USED TO SEPARATE THE SEARCH STRINGS IN THE S AND J COMMANDS, AND IS USED AT THE END OF THE COMMANDS IF ADDITIONAL COMMANDS FOLLOW. FOR EXAMPLE, THE FOLLOWING COMMAND SEQUENCE SEARCHES FOR THE STRING 'GAMMA', SUBSTITUTES THE STRING 'DELTA', AND THEN TYPES THE FIRST PART OF THE LINE WHERE THE CHANGE OCCURRED, FOLLOWED BY THE REMAINDER OF THE LINE WHICH WAS CHANGED:

SGAMMA<ENDFILE>DELTA<ENDFILE>OTT<CR>

THE CONTROL-L CHARACTER IN SEARCH AND SUBSTITUTE STRINGS IS REPLACED ON INPUT BY <CR><LF> CHARACTERS. THE CONTROL-I KEY IS TAKEN AS A TAB CHARACTER.

THE COMMANDS R & X MUST BE FOLLOWED BY A FILE NAME (WITH default FILE TYPE OF 'LIB') WITH A TRAILING <CR> OR <ENDFILE>. THE COMMAND I IS FOLLOWED BY A STRING OF SYMBOLS TO INSERT, TERMINATED BY A <CR> OR <ENDFILE>. IF SEVERAL LINES OF TEXT ARE TO BE INSERTED, THE I CAN BE DIRECTLY FOLLOWED BY AN <ENDFILE> OR <CR> IN WHICH CASE THE EDITOR ACCEPTS LINES OF INPUT TO THE NEXT <ENDFILE>. THE COMMAND OT PRINTS THE FIRST PART OF THE CURRENT LINE, AND THE COMMAND OL MOVES THE REFERENCE TO THE BEGINNING OF THE CURRENT LINE. THE COMMAND OP PRINTS THE CURRENT PAGE ONLY, WHILE THE COMMAND OZ READS THE CONSOLE RATHER THAN WAITING (THIS IS USED AGAIN WITHIN MACROS TO STOP THE DISPLAY - THE MACRO EXPANSION STOPS UNTIL A CHARACTER IS READ. IF THE CHARACTER IS NOT A BREAK THEN THE MACRO EXPANSION CONTINUES NORMALLY).

NOTE THAT A POUND SIGN IS TAKEN AS THE NUMBER 65535, ALL UNSIGNED NUMBERS ARE ASSUMED POSITIVE, AND A SINGLE - IS ASSUMED -1

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. #

A NUMBER OF COMMANDS CAN BE GROUPED TOGETHER AND EXECUTED REPETITIVELY USING THE MACRO COMMAND WHICH TAKES THE FORM

<NUMBER>MC1C2...CN<DELIMITER>

WHERE <NUMBER> IS A NON-NEGATIVE INTEGER N, AND <DELIMITER> IS <ENDFILE> OR <CR>. THE COMMANDS C1 ... CN FOLLOWING THE M ARE EXECUTED N TIMES, STARTING AT THE CURRENT POSITION IN THE BUFFER. IF N IS 0, 1, OR OMITTED, THE COMMANDS ARE EXECUTED UNTIL THE END OF THE BUFFER IS ENCOUNTERED.

THE FOLLOWING MACRO, FOR EXAMPLE, CHANGES ALL OCCURRENCES OF THE NAME 'GAMMA' TO 'DELTA', AND PRINTS THE LINES WHICH WERE CHANGED:

MFGAMMA<ENDFILE>-SDIDELTA<ENDFILE>OLT<CR>

(NOTE: AN <ENDFILE> IS THE CP/M END OF FILE MARK - CONTROL-Z)

IF ANY KEY IS DEPRESSED DURING TYPING OR MACRO EXPANSION, THE FUNCTION IS CONSIDERED TERMINATED, AND CONTROL RETURNS TO THE OPERATOR.

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. #

ERROR CONDITIONS ARE INDICATED BY PRINTING ONE OF THE CHARACTERS:

| SYMBOL   | ERROR CONDITION                                                                                   |
|----------|---------------------------------------------------------------------------------------------------|
| GREATER  | FREE MEMORY IS EXHAUSTED - ANY COMMAND CAN BE ISSUED WHICH DOES NOT INCREASE MEMORY REQUIREMENTS. |
| QUESTION | UNRECOGNIZED COMMAND OR ILLEGAL NUMERIC FIELD                                                     |
| POUND    | CANNOT APPLY THE COMMAND THE NUMBER OF TIMES SPECIFIED (OCCURS IF SEARCH STRING CANNOT BE FOUND)  |
| LETTER O | CANNOT OPEN <FILENAME>.LIB IN R COMMAND                                                           |

THE ERROR CHARACTER IS ALSO ACCOMPANIED BY THE LAST CHARACTER SCANNED WHEN THE ERROR OCCURRED. \*/

```
$ eject
```

```
/* ****
```

```
*** GLOBAL VARIABLES ***
```

```
*****
```

```
24 1  DECLARE LIT LITERALLY 'LITERALLY',
      DCL LIT 'DECLARE',
      PROC LIT 'PROCEDURE',
      ADDR LIT 'ADDRESS',
      BOOLEAN LIT 'BYTE',
      CTLL LIT '0CH',
      CTLR LIT '12H',          /* REPEAT LINE IN INSERT MODE */
      CTLU LIT '15H',          /* LINE DELETE IN INSERT MODE */
      CTLX LIT '18H',          /* EQUIVALENT TO CTLU */
      CTLH LIT '08H',          /* BACKSPACE */
      TAB LIT '09H',           /* TAB CHARACTER */
      LCA LIT '110$0001B',      /* LOWER CASE A */
      LCZ LIT '111$1010B',      /* LOWER CASE Z */
      ESC LIT '1BH',           /* ESCAPE CHARACTER */
      ENDFILE LIT '1AH';       /* CP/M END OF FILE */
```

```
25 1  DECLARE
      TRUE LITERALLY '1',
      FALSE LITERALLY '0',
      FOREVER LITERALLY 'WHILE TRUE',
      CTRL$Y LITERALLY '19h',
      CR LITERALLY '13',
      LF LITERALLY '10',
      WHAT LITERALLY '63';
```

```
26 1  DECLARE
      MAX ADDRESS,             /* .MEMORY(MAX)=0 (END) */
      MAXM ADDRESS,            /* MINUS 1 */
      HMAX ADDRESS;            /* = MAX/2 */
```

```
27 1  declare
      i byte;                  /* used by command parsing */
```

```
28 1  DECLARE
      us literally '8',         /* file from user 0 */
      RO LITERALLY '9',         /* R/O FILE INDICATOR */
      SY LITERALLY '10',        /* SYSTEM FILE ATTRIBUTE */
      EX LITERALLY '12',        /* EXTENT NUMBER POSITION */
      UB LITERALLY '13',        /* UNFILLED BYTES */
      ck LITERALLY '13',        /* checksum */
      MD LITERALLY '14',        /* MODULE NUMBER POSITION */
      NR LITERALLY '32',        /* NEXT RECORD FIELD */
      FS LITERALLY '33',        /* FCB SIZE */
      RFCB (FS) BYTE            /* READER FILE CONTROL BLOCK */
      INITIAL(0, /* FILE NAME */ ' ',
               /* FILE TYPE */ 'LIB',0,0,0),
      RBP BYTE,                 /* READ BUFFER POINTER */
```

COPYRIGHT ©1982  
DIGITAL RESEARCH  
P.O. BOX 579  
PACIFIC GROVE, CA 93950

SER. # \_\_\_\_\_

```

XFCB (FS) BYTE          /* XFER FILE CONTROL BLOCK */
    INITIAL(0, 'X$$$$$$', 'LIB', 0,0,0,0,0,0,0),
XFCBE BYTE AT(.XFCB(EX)), /* XFCB EXTENT */
XFCBR BYTE AT(.XFCB(NR)), /* XFCB RECORD # */
xfcbebt byte initial(0),  /* save xfcbe extent for appends */
xfcbrct byte initial(0),  /* save xfcbe record for appends */
XBUFF (SECTSIZE) BYTE,    /* XFER BUFFER */
XBP BYTE,                 /* XFER POINTER */

```

```

NBUF BYTE,               /* NUMBER OF BUFFERS */
BUFFLENGTH ADDRESS,      /* NBUF * SECTSIZE */
SFCB (FS) BYTE AT(.FCB), /* SOURCE FCB = DEFAULT FCB */
SDISK BYTE AT (.FCB),    /* SOURCE DISK */
SBUFFADR ADDRESS,        /* SOURCE BUFFER ADDRESS */
SBUFF BASED SBUFFADR (128) BYTE, /* SOURCE BUFFER */
password (16) byte initial(0), /* source password */

```

COPYRIGHT ©1982  
DIGITAL RESEARCH  
P.O. BOX 579  
PACIFIC GROVE, CA 93950

SER. #

```

DFCB (FS) BYTE,          /* DEST FILE CONTROL BLOCK */
DDISK BYTE AT (.DFCB),   /* DESTINATION DISK */
DBUFFADR ADDRESS,        /* DESTINATION BUFFER ADDRESS */
DBUFF BASED DBUFFADR (128) BYTE, /* DEST BUFFER */
NSOURCE ADDRESS,         /* NEXT SOURCE CHARACTER */
NDEST ADDRESS,           /* NEXT DESTINATION CHAR */

```

```

tmpfcb (FS) BYTE;        /* temporary fcb for rename & deletes */

```

```

29 1  DECLARE          /**** some of the logicals ****/
    newfile    BOOLEAN initial (false), /* true if no source file */
    onefile    BOOLEAN initial (true),  /* true if output file=input file */
    XFERON     BOOLEAN initial (false), /* TRUE IF XFER ACTIVE */
    reading    BOOLEAN initial (false), /* TRUE IF reading RFCB */
    PRINTSUPPRESS BOOLEAN initial (false), /* TRUE IF PRINT SUPPRESSED */
    sys        BOOLEAN initial (false), /* true if system file */
    protection BOOLEAN initial (false), /* password protection mode */
    INSERTING  BOOLEAN,                /* TRUE IF INSERTING CHARACTERS */
    READBUFF   BOOLEAN,                /* TRUE IF END OF READ BUFFER */
    TRANSLATE  BOOLEAN initial (false), /* TRUE IF XLATION TO UPPER CASE */
    UPPER      BOOLEAN initial (false), /* TRUE IF GLOBALLY XLATING TO UC */
    LINESET    BOOLEAN initial (true),  /* TRUE IF LINE #'S PRINTED */
    hasbdos3   BOOLEAN initial (false), /* true if BDOS version >= 3.0 */
    tail       BOOLEAN initial (true),  /* true if reading from cmd tail */
    dot$found  BOOLEAN initial (false); /* true if dot found in fname parse*/

```

```

30 1  DECLARE
    dtype (3) byte,          /* destination file type */
    libfcb (12) byte initial(0, 'X$$$$$$LIB'), /* default lib name */
    tempfl (3) byte initial('$$$'), /* temporary file type */
    backup (3) byte initial('BAK'); /* backup file type */

```

```

31 1  declare
    error$code address;

```

```

32 1  DECLARE
    COLUMN BYTE initial(0), /* CONSOLE COLUMN POSITION */
    SCOLUMN BYTE INITIAL(8), /* STARTING COLUMN IN 'I' MODE */
    TCOLUMN BYTE,           /* TEMP DURING BACKSPACE */
    CCOLUMN BYTE;          /* TEMP DURING BACKSPACE */

```



```

33 1  DECLARE DCNT BYTE;          /* RETURN CODE FROM MON? CALLS */

    /* COMMAND BUFFER */
34 1  DECLARE (MAXLEN,COMLEN) BYTE, COMBUFF(128) BYTE,
    CBP BYTE initial(0);

35 1  DECLARE /* LINE COUNTERS */
    BASELINE ADDRESS,          /* CURRENT LINE */
    RELLINE ADDRESS;          /* RELATIVE LINE IN TYPEOUT */

36 1  DECLARE
    FORWARD LIT '1',
    BACKWARD LIT '0',
    RUBOUT LIT '07FH',
    POUND LIT '23H',
    MACSIZE LIT '128',          /* MAX MACRO SIZE */
    SCRFSIZE LIT '100',        /* SCRATCH BUFFER SIZE */
    CONSIZE LIT 'ADDRESS';     /* DETERMINES MAX COMMAND NUMBER*/

37 1  DCL MACRO(MACSIZE) BYTE,
    SCRATCH(SCRFSIZE) BYTE,    /* SCRATCH BUFFER FOR F,N,S */
    (WBP, WBE, WBJ) BYTE,     /* END OF F STRING, S STRING, J STRING */
    (FLAG, MP, MI, XP) BYTE,
    MT CONSIZE;

38 1  DCL (START, RESTART, OVERCOUNT, OVERFLOW,
    disk$err, dir$err, RESET, BADCOM) LABEL;

    /* global variables used by file parsing routines */
39 1  dcl ncmd byte initial(0);

40 1  DCL (DISTANCE, TDIST) CONSIZE,
    (DIRECTION, CHAR) BYTE,
    ( FRONT, BACK, FIRST, LASTC) ADDR;

41 1  dcl LPP byte initial(23);    /* LINES PER PAGE */

    /* the following stucture is used near plm: to set
    the lines per page from the BDOS 3 SCB */
42 1  declare
    pb (2) byte data (28,0);

43 1  declare
    ver address;                /* VERSION NUMBER */

44 1  declare
    err$msg          address initial(0),
    invalid (*)      byte data ('Invalid Filename$'),
    dirfull (*)      byte data ('DIRECTORY FULL$'),
    diskfull (*)     byte data ('DISK FULL$'),
    password$err(*)  byte data ('Creating Password$'),
    not$found (*)    byte data ('File not found$'),
    not$avail (*)    byte data ('File not available$');

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. #

\$ eject

/\* \*\*\*\* \*/

\*\*\* CP/M INTERFACE ROUTINES \*\*\*

\*\*\*\*\*/

/\* IO SECTION \*/

```

45 1  READCHAR: PROCEDURE BYTE; RETURN MON2(1,0);
47 2      END READCHAR;

48 1      conin;
        procedure byte;
49 2      return mon2(6,0fdh);
50 2      end conin;

51 1  PRINTCHAR: PROCEDURE(CHAR);
52 2      DECLARE CHAR BYTE;
53 2      IF PRINTSUPPRESS THEN RETURN;
55 2      CALL MON1(2,CHAR);
56 2      END PRINTCHAR;

57 1  TTYCHAR: PROCEDURE(CHAR);
58 2      DECLARE CHAR BYTE;
59 2      IF CHAR >= ' ' THEN COLUMN = COLUMN + 1;
61 2      IF CHAR = LF THEN COLUMN = 0;
63 2      CALL PRINTCHAR(CHAR);
64 2      END TTYCHAR;

65 1  BACKSPACE: PROCEDURE;
        /* MOVE BACK ONE POSITION */
66 2      IF COLUMN = 0 THEN RETURN;
68 2      CALL TTYCHAR(CTLH); /* COLUMN = COLUMN - 1 */
69 2      CALL TTYCHAR(' '); /* COLUMN = COLUMN + 1 */
70 2      CALL TTYCHAR(CTLH); /* COLUMN = COLUMN - 1 */
71 2      COLUMN = COLUMN - 2;
72 2      END BACKSPACE;

73 1  PRINTABS: PROCEDURE(CHAR);
74 2      DECLARE (CHAR,I,J) BYTE;
75 2      I = CHAR = TAB AND 7 - (COLUMN AND 7);
76 2      IF CHAR = TAB THEN CHAR = ' ';
78 2      DO J = 0 TO I;
79 3          CALL TTYCHAR(CHAR);
80 3          END;
81 2      END PRINTABS;

82 1  GRAPHIC: PROCEDURE(C) BOOLEAN;
83 2      DECLARE C BYTE;
        /* RETURN TRUE IF GRAPHIC CHARACTER */
84 2      IF C >= ' ' THEN RETURN TRUE;
86 2      RETURN C = CR OR C = LF OR C = TAB;

```

COPYRIGHT ©1982  
DIGITAL RESEARCH  
P.O. BOX 579  
PACIFIC GROVE, CA 93950

SER. # \_\_\_\_\_

```

87 2      END GRAPHIC;

88 1      PRINTC: PROCEDURE(C);
89 2          DECLARE C BYTE;
90 2          IF NOT GRAPHIC(C) THEN
91 2              DO; CALL PRINTABS('^');
93 3              C = C + '0';
94 3              END;
95 2          CALL PRINTABS(C);
96 2          END PRINTC;

97 1      CRLF: PROCEDURE;
98 2          CALL PRINTC(CR); CALL PRINTC(LF);
100 2      END CRLF;

101 1     PRINTM: PROCEDURE(A);
102 2         DECLARE A ADDRESS;
103 2         CALL MON1(9,A);
104 2         END PRINTM;

105 1     PRINT: PROCEDURE(A);
106 2         DECLARE A ADDRESS;
107 2         CALL CRLF;
108 2         CALL PRINTM(A);
109 2         END PRINT;

110 1     perror: procedure(a);
111 2         declare a address;
112 2         call print(,(tab,'ERROR - $'));
113 2         call printm(A);
114 2         call crlf;
115 2         end perror;

116 1     READ: PROCEDURE(A);
117 2         DECLARE A ADDRESS;
118 2         CALL MON1(10,A);
119 2         END READ;

/* used for library files */
120 1     OPEN: PROCEDURE(FCB);
121 2         DECLARE FCB ADDRESS;
122 2         if MON2(15,FCB) = 255 then do;
124 3             flag = '0';
125 3             err$msg = .not$found;
126 3             go to reset;
127 3             end;
128 2         END OPEN;

/* used for main source file */
129 1     OPEN$FILE: PROCEDURE(FCB) ADDRESS;
130 2         DECLARE FCB ADDRESS;
131 2         RETURN MON3(15,FCB);
132 2         END OPEN$FILE;

133 1     CLOSE: PROCEDURE(FCB);
134 2         DECLARE FCB ADDRESS;
135 2         BCNT = MON2(16,FCB);

```

COPYRIGHT ©1982  
 DIGITAL RESEARCH  
 P. O. BOX 579  
 PACIFIC GROVE, CA 93950

SER. # \_\_\_\_\_

```

136 2      END CLOSE;

137 1      DELETE: PROCEDURE(FCB);
138 2      DECLARE FCB ADDRESS;
139 2      DCNT = MON2(19,FCB);
140 2      END DELETE;

141 1      DISKREAD: PROCEDURE(FCB) BYTE;
142 2      DECLARE FCB ADDRESS;
143 2      RETURN MON2(20,FCB);
144 2      END DISKREAD;

145 1      DISKWRITE: PROCEDURE(FCB) BYTE;
146 2      DECLARE FCB ADDRESS;
147 2      RETURN MON2(21,FCB);
148 2      END DISKWRITE;

149 1      RENAME: PROCEDURE(FCB);
150 2      DECLARE FCB ADDRESS;
151 2      CALL MON1(23,FCB);
152 2      END RENAME;

153 1      READCOM: PROCEDURE;
154 2      MAXLEN = 128; CALL READ(.MAXLEN);
156 2      END READCOM;

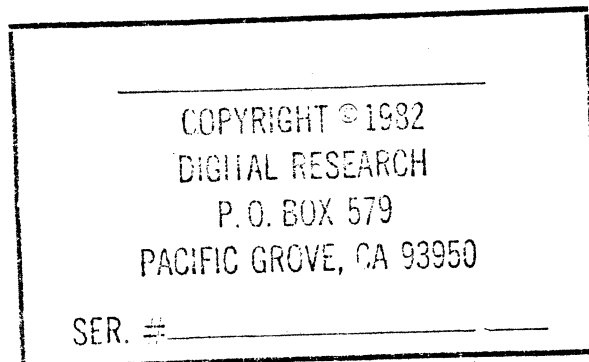
157 1      BREAK$KEY: PROCEDURE BOOLEAN;
158 2      IF MON2(11,0) THEN
159 2          DO; /* CLEAR CHAR */
160 3          IF MON2(1,0) = CTRL$Y THEN
161 3              RETURN TRUE;
162 3          END;
163 2      RETURN FALSE;
164 2      END BREAK$KEY;

165 1      CSELECT: PROCEDURE BYTE;
          /* RETURN CURRENT DRIVE NUMBER */
166 2      RETURN MON2(25,0);
167 2      END CSELECT;

168 1      SETDMA: PROCEDURE(A);
169 2      DECLARE A ADDRESS;
          /* SET DMA ADDRESS */
170 2      CALL MON1(26,A);
171 2      END SETDMA;

172 1      set$attribute: procedure(FCB);
173 2      declare fcb address;
174 2      call MON1(30,FCB);
175 2      end set$attribute;

```



/\* The PL/M built-in procedure "MOVE" can be used to move storage,  
its definition is:

```

MOVE: PROCEDURE(COUNT,SOURCE,DEST);
      DECLARE (COUNT,SOURCE,DEST) ADDRESS;
      / MOVE DATA FROM SOURCE TO DEST ADDRESSES, FOR COUNT BYTES /

```

```

      END MOVE;
      */
      /* this routine is included solely for
         economy of space over the use of the
         equivalent (in-line) code generated by
         the built-in function */
176 1  move:  proc(c,s,d);
177 2      dcl (s,d) addr, c byte;
178 2      dcl a based s byte, b based d byte;

179 2      do while (c:=c-1)<>255;
180 3          b=a; s=s+1; d=d+1;
183 3          end;
184 2      end move;

185 1  write$xfcb: PROCEDURE(FCB);
186 2      DECLARE FCB ADDRESS;
187 2      call move(8,password,password(8));
188 2      if MON2(103,FCB)= Offh then
189 2          call perror(password$err);
190 2      END write$xfcb;

191 1  read$xfcb: PROCEDURE(FCB);
192 2      DECLARE FCB ADDRESS;
193 2      call MON1(102,FCB);
194 2      END read$xfcb;

      /* Off => return BDOS errors */
195 1  return$errors:
      procedure(mode);
196 2      declare mode byte;
197 2      call mon1 (45,mode);
198 2      end return$errors;

199 1  REBOOT: PROCEDURE;
200 2      IF XFERON THEN
201 2          CALL DELETE(.libfcb);
202 2          CALL BOOT;
203 2          END REBOOT;

204 1  version: procedure address;
      /* returns current cp/m version # */
205 2      return mon3(12,0);
206 2      end version;

```

COPYRIGHT ©1982  
 DIGITAL RESEARCH  
 P.O. BOX 579  
 PACIFIC GROVE, CA 93950

SER. # \_\_\_\_\_

```
$ eject
```

```
/* ****
```

```
*** SUBROUTINES ***
```

```
***** */
```

```
/* INPUT / OUTPUT BUFFERING ROUTINES */
```

```
/* abort ED and print error message */
```

```
207 1  ABORT: PROCEDURE(A);
208 2    DECLARE A ADDRESS;
209 2    CALL perror(A);
210 2    CALL REBOOT;
211 2    END ABORT;
```

```
/* ----- */
```

```
/* fatal file error */
```

```
212 1  FERR: PROCEDURE;
213 2    CALL CLOSE(.DFCB); /* ATTEMPT TO CLOSE FILE FOR LATER RECOVERY */
214 2    CALL ABORT (.dirfull);
215 2    END FERR;
```

```
/* ----- */
```

```
/* set password if cpm 3 */
```

```
216 1  setpassword: procedure;
217 2    if has$bdos3 then
218 2        call setdma(.password);
219 2    end setpassword;
```

```
/* ----- */
```

```
/* delete file at afcb */
```

```
220 1  delete$file: procedure(afcb);
221 2    declare afcb address;
222 2    call setpassword;
223 2    call delete(afcb);
224 2    end delete$file;
```

```
/* ----- */
```

```
/* rename file at afcb */
```

```
225 1  rename$file: procedure(afcb);
226 2    declare afcb address;
227 2    call delete$file(afcb+16); /* delete new file */
228 2    call setpassword;
229 2    call rename(afcb);
230 2    end rename$file;
```

```
/* ----- */
```

COPYRIGHT © 1982  
DIGITAL RESEARCH  
P. O. BOX 579  
PACIFIC GROVE, CA 93950

SER. # \_\_\_\_\_

```
                /* make file at afcb */
231 1  make$file: procedure(afcb);
232 2      declare afcb address;
233 2      call delete$file(afcb); /* delete file */
234 2      call setpassword;
235 2      DCNT = MON2(22,afcb); /* create file */
236 2      end make$file;
/* ----- */
```

```
                /* fill string @ s for c bytes with f */
237 1  fill: proc(s,f,c);
238 2      dcl s addr,
          (f,c) byte,
          a based s byte;

239 2      do while (c:=c-1)<>255;
240 3          a = f;
241 3          s = s+1;
242 3          end;
243 2      end fill;
```

COPYRIGHT ©1982  
DIGITAL RESEARCH  
P.O. BOX 579  
PACIFIC GROVE, CA 93950

SER. # \_\_\_\_\_

\$ eject

/\* \*\*\*\*

\*\*\* FILE HANDLING ROUTINES \*\*\*

\*\*\*\*\*/

/\* set destination file type to type at A \*/

```
244 1 SETTYPE: PROCEDURE(afcb,A);
245 2     DECLARE (afcb, A) ADDRESS;
246 2     CALL MOVE(3,A,aFCB+9);
247 2     END SETTYPE;
/* ----- */
```

/\* set dma to xfer buffer \*/

```
248 1 SETXDMA: PROCEDURE;
249 2     CALL SETDMA(.XBUF);
250 2     END SETXDMA;
/* ----- */
```

/\* fill primary source buffer \*/

```
251 1 FILLSOURCE: PROCEDURE;
252 2     DECLARE I BYTE;
253 2     ZN: PROCEDURE;
254 3         NSOURCE = 0;
255 3         END ZN;

256 2     CALL ZN;
257 2         DO I = 0 TO NBUF;
258 3             CALL SETDMA(SBUFFADR+NSOURCE);
259 3             IF (DCNT != DISKREAD(.FCB)) <> 0 THEN
260 4                 DO; IF DCNT > 1 THEN CALL FERR;
261 4                 SBUFF(NSOURCE) = ENDFILE;
262 4                 I = NBUF;
263 4                 END;
264 4             ELSE
265 4                 NSOURCE = NSOURCE + SECTSIZE;
266 3             END;
267 3             CALL ZN;
268 2         END FILLSOURCE;
269 2     END FILLSOURCE;
/* ----- */
```

/\* get next character in source file \*/

```
270 1 GETSOURCE: PROCEDURE BYTE;
271 2     DECLARE B BYTE;
272 2     if newfile then return endfile; /* in case they try to to */
273 2     IF NSOURCE >= BUFLNGTH THEN CALL FILLSOURCE;
274 2     IF (B != SBUFF(NSOURCE)) <> ENDFILE THEN
275 2         NSOURCE = NSOURCE + 1;
```

COPYRIGHT ©1982  
DIGITAL RESEARCH  
P.O. BOX 579  
PACIFIC GROVE, CA 93950

SER. # \_\_\_\_\_



```

278 2      RETURN B;
279 2      END GETSOURCE;
/* ----- */

```

```

280 1      /* try to free space by erasing backup */
      erase$bak: PROCEDURE BOOLEAN;

281 2          if onefile then
282 2              if newfile then do;
284 3                  call move(fs,dfcb,.tapfcb); /* can't diddle with open fcb */
285 3                  CALL SETTYPE(.tapfcb,.BACKUP);
286 3                  CALL DELETE$file(.tapfcb);
287 3                  if dcnt < 255 then
288 3                      return true;
289 3                  end;
290 2          return false;
291 2      end erase$bak;

```

```

/* ----- */

```

```

      /* write output buffer up to (not including)
      ndest (low 7 bits of ndest are 0 */

292 1      WRITEDEST: PROCEDURE;
293 2          DECLARE (I,N,save$ndest) BYTE;

294 2          n = shr(ndest,sectshf); /* calculate number sectors to write */
295 2          if n=0 then return; /* no need to write if we haven't filled sector */
297 2          save$ndest = ndest; /* save for error recovery */
298 2          ndest = 0;
299 2          DO I = 1 TO N;
300 3      retry:
          CALL SETDMA(DBUFFADR+NDEST);
          IF DISKWRITE(.DFCB) < 0 THEN
302 3              if erase$bak then
303 3                  go to retry;
304 3              else do; /* reset buffer, let them take action (delete files) */
305 4                  if ndest < 0 then
306 4                      call move(save$ndest-ndest, dbuffadr+ndest, dbuffadr);
307 4                      ndest = save$ndest-ndest;
308 4                      go to disk$err;
309 4                  end;
310 3              NDEST = NDEST + SECTSIZE;
311 3              END;
312 2          ndest = 0;
313 2          END WRITEDEST;
/* ----- */

```

```

      /* put a character in output buffer */

314 1      PUTDEST: PROCEDURE(B);
315 2          DECLARE B BYTE;
316 2          IF NDEST >= BUFFLENGTH THEN CALL WRITEDEST;
318 2          DBUFF(NDEST) = B;
319 2          NDEST = NDEST + 1;

```

COPYRIGHT ©1982  
 DIGITAL RESEARCH  
 P.O. BOX 579  
 PACIFIC GROVE, CA 93950  
 SER. # \_\_\_\_\_

```

320 2      END PUTDEST;
/* ----- */

```

```

/* put a character in the xfer buffer */
321 1      PUTXFER: PROCEDURE(C);
322 2      DECLARE C BYTE;
323 2      IF XBP >= SECTSIZE THEN /* BUFFER OVERFLOW */
324 2          DO;
325 3      retry:
          CALL SETXDMA;
326 3          xfcbe = xfcbe; /* save for appends */
327 3          xfcbr = xfcbr;
328 3          IF DISKWRITE(.XFCB) < 0 THEN
329 3              if erasebak then
330 3                  go to retry;
331 3              else do;
          /****** call close(.xfcb); *** commented out whf 8/82 !!!! *****/
332 4              go to diskerr;
333 4              end;
334 3          XBP = 0;
335 3          END;
336 2          XBUFF(XBP) = C; XBP = XBP + 1;
338 2      END PUTXFER;
/* ----- */

```

COPYRIGHT © 1982  
DIGITAL RESEARCH  
P.O. BOX 579  
PACIFIC GROVE, CA 93950

SER. # \_\_\_\_\_

```

/* empty xfer buffer and close file.
   This routine is added to allow saving lib
   files for future edits - DH 10/18/81 */
339 1      closexfer: procedure;
340 2      dcl i byte;

341 2          do i = xbp to sectsize;
342 3          call putxfer(ENDFILE);
343 3          end;
344 2          call close(.xfcb);
345 2          end closexfer;
/* ----- */

```

```

/* compare xfer and rfer to see if same */
346 1      comparexfer: procedure BOOLEAN;
347 2      dcl i byte;

348 2          i = 12;
349 2          do while (i=i-1) < -1;
350 3          if xfer(i) < rfer(i) then
351 3              return false;
352 3          end;
353 2          return true;
354 2          end comparexfer;
/* ----- */

```

/\* restore xfer file extent and current  
record, read record and set xfer pointer

```
                                to first ENDFILE */
355 1  append$xfer; procedure;
356 2      xfcbe = xfcbe;
357 2      call open(,xfcb);
358 2      xfcbr = xfcbr;
359 2      call setxdma;
360 2      if diskread(,xfcb) = 0 then do;
361 3          xfcbr = xfcbr;          /* write same record */
362 3          do xbp = 0 to sectsize;
363 4              if xbuff(xbp) = ENDFILE then
364 5                  return;
365 4              end;
366 4          end;
367 3      end;
368 2  end append$xfer;
```

COPYRIGHT ©1982  
DIGITAL RESEARCH  
P.O. BOX 579  
PACIFIC GROVE, CA 93950  
SER. # \_\_\_\_\_

```

$ eject
/* **** */

      *** END EDIT ROUTINE ***

/* **** */

      /* finish edit, close files, rename */
369 1  FINIS: PROCEDURE;
370 2      MOVEUP: PROCEDURE(afeb);
371 3      dcl afeb address;
      /* set second filename (new name) for rename function */
372 3      CALL MOVE(16,afeb,afeb+16);
373 3      END MOVEUP;

      /* **** WRITE OUTPUT BUFFER **** */
      /* SET UNFILLED BYTES - USED FOR ISIS-II COMPATIBILITY */
      /* DFUB = 0 ; <<<< REMOVE FOR MP/M 2 , CP/M 3 */
374 2      DO WHILE (LOW(NDST) AND 7FH) <> 0;
      /* COUNTS UNFILLED BYTES IN LAST RECORD */
      /* DFUB = DFUB + 1; */
375 3      CALL PUTDEST(ENDFILE);
376 3      END;
377 2      CALL WRITEDST;

      if not newfile then
378 2          call close(.sfcb); /* close this to clean up for mp/m environs */
379 2

      /* **** CLOSE TEMPORARY DESTINATION FILE **** */
380 2      CALL CLOSE(.DFCB);
381 2      IF DCNT = 255 THEN CALL FERR;
383 2      if sys then do;
385 3          dfcb(sy)=dfcb(sy) or 80h;
386 3          call setpassword;
387 3          call set$attribute(.dfcb);
388 3          end;

      /* **** RENAME SOURCE TO BACKUP IF ONE FILE **** */
389 2      if onefile then do;
391 3          call moveup(.sfcb);
392 3          CALL SETTYPE(.sfcb+16,.BACKUP); /* set new type to BAK */
393 3          CALL RENAME$FILE(.SFCB);
394 3          end;

      /* **** RENAME TEMPORARY DESTINATION FILE **** */
395 2      CALL MOVEUP(.DFCB);
396 2      CALL SETTYPE(.DFCB+16,.DTYPE);
397 2      CALL RENAME$FILE(.DFCB);

398 2      END FINIS;

```

COPYRIGHT ©1982  
 DIGITAL RESEARCH  
 P.O. BOX 579  
 PACIFIC GROVE, CA 93950

SER. # \_\_\_\_\_

\$ eject

/\* \*\*\*\*

\*\*\* COMMAND ROUTINES \*\*\*

\*\*\*\*\*/

/\* print a character if not macro expansion \*/

```
399 1 PRINTMAC: PROCEDURE(CHAR);
400 2     DECLARE CHAR BYTE;
401 2     IF MP <> 0 THEN RETURN;
403 2     CALL PRINTC(CHAR);
404 2     END PRINTMAC;
/* ----- */
```

/\* return true if lower case character \*/

```
405 1 LOWERCASE: PROCEDURE(C) BOOLEAN;
406 2     DECLARE C BYTE;
407 2     RETURN C >= LCA AND C <= LCZ;
408 2     END LOWERCASE;
/* ----- */
```

/\* translate character to upper case \*/

```
409 1 UCASE: PROCEDURE(C) BYTE;
410 2     DECLARE C BYTE;
411 2     IF LOWERCASE(C) THEN RETURN C AND SFH;
413 2     RETURN C;
414 2     END UCASE;
/* ----- */
```

/\* get password and place at fcb + 16 \*/

```
415 1 getpasswd: proc;
416 2     dcl (i,c) byte;

417 2     call crlf;
418 2     call print(,('Password ? ', '$'));
419 2     retry:
420 2         call fill(,password,' ',8);
421 3     nxtchr:
422 3         if (c:=ucase(conin)) >= ' ' then
423 3             password(i)=c;
424 3         if c = cr then
425 3             go to exit;
426 3         if c = CTLX then
427 3             goto retry;
428 3         if c = CTLH then do;
429 4             if i<1 then
430 4                 goto retry;
```

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # \_\_\_\_\_

```

431 4      else do;
432 5          password(i:=i-1)=' ';
433 5          goto nxtchr;
434 5          end;
435 4      end;
436 3      if c = 3 then
437 3          call reboot;
438 3      end;
439 2  exit:
      c = break$Key;    /* clear raw I/O mode */
440 2      end getpasswd;
/* ----- */

```

```

/* translate to upercase if translate flag
is on (also translate ESC to ENDFILE) */
441 1  UTRAN: PROCEDURE(C) BYTE;
442 2      DECLARE C BYTE;
443 2      IF C = ESC THEN C = ENDFILE;
445 2      IF TRANSLATE THEN RETURN UCASE(C);
447 2      RETURN C;
448 2      END UTRAN;
/* ----- */

```

```

/* print the line number */
449 1  PRINTVALUE: PROCEDURE(V);
      /* PRINT THE LINE VALUE V */
450 2      DECLARE D BYTE,
      ZERO BOOLEAN,
      (K,V) ADDRESS;
451 2      K = 10000;
452 2      ZERO = FALSE;
453 2      DO WHILE K <> 0;
454 3          D = LOW(V/K); V = V MOD K;
456 3          K = K / 10;
457 3          IF ZERO OR D <> 0 THEN
458 3              DO; ZERO = TRUE;
460 4              CALL PRINTC('0'+D);
461 4              END;
      ELSE
462 3          CALL PRINTC(' ');
463 3      END;
464 2      END PRINTVALUE;
/* ----- */

```

COPYRIGHT © 1982  
 DIGITAL RESEARCH  
 P.O. BOX 579  
 PACIFIC GROVE, CA 93950

SER. # \_\_\_\_\_

```

/* print line with number V */
465 1  PRINTLINE: PROCEDURE(V);
466 2      DECLARE V ADDRESS;
467 2      IF NOT LINESET THEN RETURN;
469 2      CALL PRINTVALUE(V);
470 2      CALL PRINTC(':');
471 2      CALL PRINTC(' ');
472 2      IF INSERTING THEN
473 2          CALL PRINTC(' ');
      ELSE

```

```

474 2      CALL PRINTC('*');
475 2      END PRINTLINE;
/* ----- */

/* print current line (baseline) */
476 1  PRINTBASE: PROCEDURE;
477 2      CALL PRINTLINE(BASELINE);
478 2      END PRINTBASE;
/* ----- */

/* print current line if not in a macro */
479 1  PRINTNMBASE: PROCEDURE;
480 2      IF MP <> 0 THEN RETURN;
482 2      CALL PRINTBASE;
483 2      END PRINTNMBASE;
/* ----- */

/* get next character from command tail */
484 1  getcmd: proc byte;
485 2      if buff(ncmd+1) <> 0 then
486 2          return buff(ncmd := ncmd + 1);
487 2      return cr;
488 2      end getcmd;
/* ----- */

/* read next char from command buffer */
489 1  READC: PROCEDURE BYTE;
/* MAY BE MACRO EXPANSION */
490 2      IF MP > 0 THEN
491 2          DO;
492 3              IF BREAK$KEY THEN GO TO OVERCOUNT;
494 3              IF XP >= MP THEN
495 3                  DO; /* START AGAIN */
496 4                      IF MT <> 0 THEN
497 4                          DO; IF (MT:=MT-1) = 0 THEN
499 5                              GO TO OVERCOUNT;
500 5                          END;
501 4                          XP = 0;
502 4                          END;
503 3                      RETURN UTRAN(MACRO((XP := XP + 1) - 1));
504 3                      END;
505 2              IF INSERTING THEN RETURN UTRAN(READCHAR);

/* GET COMMAND LINE */
507 2      IF READBUFF THEN
508 2          DO; READBUFF = FALSE;
510 3              IF LINESET AND COLUMN = 0 THEN
511 3                  DO;
512 4                      IF BACK >= MAXN THEN
513 4                          CALL PRINTLINE(0);
ELSE
514 4                      CALL PRINTBASE;
515 4                      END;

```

COPYRIGHT ©1982  
 DIGITAL RESEARCH  
 P. O. BOX 579  
 PACIFIC GROVE, CA 93950  
 SER. # \_\_\_\_\_

```

ELSE
516 3      CALL PRINTC('*');
517 3      CALL READCOM; CBP = 0;
519 3      CALL PRINTC(LF);
520 3      COLUMN = 0;
521 3      END;
522 2      IF (READBUFF != CBP = COMLEN ) THEN
523 2          COMBUFF(CBP) = CR;
524 2      RETURN UTRAN(COMBUFF((CBP := CBP +1) -1));
525 2      END READC;

/* ----- */

/* get upper case character from command
   buffer or command line */
526 1      get$uc: proc;
527 2          if tail then
528 2              char = ucase(getcmd);
           else
529 2              char = ucase(readc);
530 2          end get$uc;

/* ----- */

/* parse file name
   this routine requires a routine to get
   the next character and put it in a byte
   variable */
531 1      parse$fcb: proc(fcbadr) byte;
532 2          dcl fcbadr addr;
533 2          dcl afcb based fcbadr (33) byte;
534 2          dcl drive lit 'afcb(0)';
535 2          dcl (i,delimiter) byte;
536 2          dcl pflag boolean;

537 2          putc: proc;
538 3              afcb(i := i + 1) = char;
539 3              pflag = true;
540 3          end putc;

541 2          delin: proc boolean;
542 3              dcl del($) byte data (CR,ENDFILE,' ,;=<>_[]*?');
                    /* 0 1 2345678901234 */
543 3              do delimiter = 0 to last(del);
544 4                  if char = del(delimiter) then do;
545 5                      if delimiter > 12 then /* * or ? */
546 5                          call perror(('Cannot Edit Wildcard Filename$'));
547 5                      return (true);
548 5                  end;
549 5                  end;
550 4              end;
551 3              return (false);
552 3          end delin;

553 2          pflag = false;
554 2          flag = true; /* global flag set to false if invalid filename */
555 2          dot$found = false; /* allow null extensions in 'parse$lib' */

```

COPYRIGHT © 1982  
 DIGITAL RESEARCH  
 P. O. BOX 579  
 PACIFIC GROVE, CA 93950

SER. # \_\_\_\_\_



```

556 2      call get$uc;
557 2      if char <> CR then
558 2          if char <> ENDFILE then do;
                    /* initialize fcb to srce fcb type & drive */
560 3          call fill(fcbadr+12,0,21);
561 3          call fill(fcbadr+1,' ',11);
                    /* clear leading blanks */
562 3          do while char = ' ';
563 4          call get$uc;
564 4          end;

                    /* parse loop */
565 3          do while not delim;
566 4          i = 0;
                    /* get name */
567 4          do while not delim;
568 5          if i > 7 then
569 5              go to err;      /* too long */
570 5          call putc;
571 5          call get$uc;
572 5          end;
573 4          if char = ':' then do;
                    /* get drive from afcb(1) */
575 5          if i < 1 then
576 5              go to err;      /* invalid : */
577 5          if (drive := afcb(1) - 'A' + 1) > 16 then
578 5              go to err;      /* invalid drive */
579 5          afcb(1) = ' ';
580 5          call get$uc;
581 5          end;
582 4          if char = '.' then do;
                    /* get file type */
584 5          i = 8;
585 5          dot$found = true;    /* .ext specified (may be null)*/
586 5          call get$uc;
587 5          do while not delim;
588 6          if i > 10 then
589 6              go to err;      /* too long */
590 6          call putc;
591 6          call get$uc;
592 6          end;
593 5          end;
594 4          if char = ';' then do;
                    /* get password */
596 5          call fill(fcbadr+16,' ',8); /* where fn #152 puts passwd */
597 5          i = 15;             /* passwd is last field */
598 5          call get$uc;
599 5          do while not delim;
600 6          if i > 23 then
601 6              go to err;
602 6          call putc;
603 6          call get$uc;
604 6          end;
605 5          call move(8,fcbadr+16,password); /* where ed wants it */
606 5          end;
607 4          end;      /* parse loop */
                    /* delimiter must be a comma or space */
608 3          if delimiter > 3 then /* not a CR,ENDFILE,SPACE,COMMA */

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # \_\_\_\_\_

```

609 3          go to err;
610 3          if not pflag then
611 3              go to err;
612 3          end;

613 2          return (pflag);

614 2  err:
        call perror(.invalid);
615 2          return (flag:=false);
616 2          end parse$fcbl;
/* ----- */

/* set up destination FCB */
617 1  setdest: PROCEDURE;
618 2          dcl i byte;

        /* onefile = true; (initialized) */
619 2          if not tail then do;
621 3              call print(.( 'Enter Output file: $' ));
622 3              call readcom;
623 3              cbp,readbuff = 0;
624 3              call crlf;
625 3              call crlf;
626 3              end;
627 2          if parse$fcbl(dfcb) then do;
629 3              onefile = false;
630 3              if dfcb(1) = ' ' then
631 3                  call move(15,.sfcbl+1,.dfcb+1);
632 3              end;
        else
633 2              CALL MOVE(16,.SFCB,.DFCB);
634 2              call move(3,.dfcb(9),.dtype); /* save destination type */
635 2              end setdest;
/* ----- */

/* set read lib file DMA address */
636 1  SETRDMA: PROCEDURE;
637 2          CALL SETDMA(.BUFF);
638 2          END SETRDMA;
/* ----- */

/* read lib file routine */
639 1  READFILE: PROCEDURE BYTE;
640 2          IF RBP >= SECTSIZE THEN
641 2              DO; CALL SETRDMA;
643 3              IF DISKREAD(.RFCB) <> 0 THEN RETURN ENDFILE;
645 3              RBP = 0;
646 3              END;
647 2          RETURN UTRAN(BUFF((RBP := RBP + 1) - 1));
648 2          END READFILE;

```

COPYRIGHT © 1982  
 DIGITAL RESEARCH  
 P. O. BOX 579  
 PACIFIC GROVE, CA 93950  
 SER. # \_\_\_\_\_

```
$ eject
```

```
/* ****
```

```
*** INITIALIZATION ***
```

```
*****/
```

```
649 1  SETUP: PROCEDURE;
```

```
/* **** OPEN SOURCE FILE **** */
```

```
650 2  sfcf(ex), sfcf(md), sfcf(nr) = 0;
651 2  if has$bdos3 then do;
653 3      call return$errors(0Feh);          /* set error mode */
654 3      call setpassword;
655 3      end;
656 2  error$code = open$file (.SFCB);
657 2  if has$bdos3 then do;                  /* extended bdos errors */
659 3      call return$errors(0);            /* reset error mode */
660 3      if low(error$code) = 0Ffh and high(error$code) = 7 then do;
662 4          call getpasswd;                /* let them enter password */
663 4          call crlf;
664 4          call crlf;
665 4          call setpassword;              /* set dma to password */
666 4          error$code = open$file(.fcb);  /* reset error$code */
667 4          end;
668 3      if low(error$code)=0Ffh and high(error$code)<>0 then
669 3          call abort(.notavail);        /* abort anything but not found */
670 3      end;
671 2  dcnt=low(error$code);
672 2  if onefile then do;
674 3      IF ROL(FCB(RO),1) THEN
675 3          CALL abort(.( 'FILE IS READ/ONLY' ));
676 3      else IF ROL(FCB(SY),1) THEN /* system attribute */
677 3          do;
678 4          if rol(FCB(us),1) then
679 4              dcnt = 255;                /* user 0 file so create */
        else
680 4          sys = true;
681 4          end;
        end;
end;
```

```
/* **** NEW FILE IF NO SOURCE FILE **** */
```

```
683 2  IF DCNT = 255 THEN do;
685 3      if not onefile then
686 3          call abort(.not$found);
687 3      newfile = true;
688 3      CALL PRINT(.( 'NEW FILE' ));
689 3      CALL CRLF;
690 3      END;
```

```
/* **** MAKE TEMPORARY DESTINATION FILE **** */
```

*CP/M Plus Ver 3.0*

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # *CP3-135*

```
691 2      CALL SETTYPE(.dfcb,.tempf1);
692 2      DFCB(EX)=0;
693 2      CALL MAKE$file(.DFCB);
694 2      if dcnt = 255 then
695 2          call ferr;
/* THE TEMP FILE IS NOW CREATED */

/* now create the password if any */
696 2      if protection <> 0 then do;
698 3          dfcb(ex) = protection or 1; /* set password */
699 3          call setpassword;
700 3          call write$xfcb(.dfcb);
701 3          end;
702 2      dfcb(ex),DFCB(32) = 0; /* NEXT RECORD IS ZERO */

/* * * * * * RESET BUFFER * * * * * */

703 2      NSOURCE = BUFLLENGTH;
704 2      NDEST = 0;
705 2      BASELINE = 1;          /* START WITH LINE 1 */
706 2      END SETUP;
```

COPYRIGHT © 1982  
DIGITAL RESEARCH  
P.O. BOX 579  
PACIFIC GROVE, CA 93950

SER. # \_\_\_\_\_

\$ eject

/\* \*\*\*\* \*/

\*\*\* BUFFER MANAGEMENT \*\*\*

\*\*\*\*\* \*/

/\* DISTANCE is the number of lines prefix  
to a command \*/

/\* set maximum distance (OFFFHH) \*/

```
707 1 SETFF: PROCEDURE;
708 2     DISTANCE = OFFFHH;
709 2     END SETFF;
/* ----- */
```

/\* return true if distance is zero \*/

```
710 1 DISTZERO: PROCEDURE BOOLEAN;
711 2     RETURN DISTANCE = 0;
712 2     END DISTZERO;
/* ----- */
```

/\* set distance to zero \*/

```
713 1 ZERODIST: PROCEDURE;
714 2     DISTANCE = 0;
715 2     END ZERODIST;
/* ----- */
```

/\* check for zero distance and decrement \*/

```
716 1 DISTNZERO: PROCEDURE BOOLEAN;
717 2     IF NOT DISTZERO THEN
718 2         DO; DISTANCE = DISTANCE - 1;
719 3         RETURN TRUE;
720 3         END;
721 3         RETURN FALSE;
722 2         END DISTNZERO;
723 2
/* ----- */
```

/\* set memory limits of command from  
distance and direction \*/

```
724 1 SETLIMITS: PROC;
725 2     DCL (I,K,L,M) ADDR, (MIDDLE,LOOPING) BYTE;
726 2     RELLINE = 1; /* RELATIVE LINE COUNT */
727 2     IF DIRECTION = BACKWARD THEN
728 2         DO; DISTANCE = DISTANCE+1; I = FRONT; L = 0; K = OFFFHH;
729 3         END;
730 3     ELSE
731 3         DO; I = BACK; L = MAXM; K = 1;
732 3         END;
```

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. #

```

739 2      LOOPING = TRUE;
740 2      DO WHILE LOOPING;
741 3          DO WHILE (MIDDLE := I <> L) AND
              MEMORY(M:=I+K) <> LF;
742 4          I = M;
743 4          END;
744 3      LOOPING = (DISTANCE := DISTANCE - 1) <> 0;
745 3      IF NOT MIDDLE THEN
746 3          DO; LOOPING = FALSE;
748 4          I = I - K;
749 4          END;
750 3      ELSE do;
751 4          RELLINE = RELLINE - 1;
752 4          IF LOOPING THEN
753 4              I = M;
754 4          end;
755 3      END;

```

```

756 2      IF DIRECTION = BACKWARD THEN
757 2          DO; FIRST = I; LASTC = FRONT - 1;
760 3          END;
761 2      ELSE
764 3          DO; FIRST = BACK + 1; LASTC = I + 1;
765 2          END;
765 2      END SETLIMITS;

```

```

/* ----- */

```

```

/* increment current position */
766 1      INCBASE: PROCEDURE;
767 2          BASELINE = BASELINE + 1;
768 2          END INCBASE;

```

```

/* ----- */

```

```

/* decrement current position */
769 1      DECBASE: PROCEDURE;
770 2          BASELINE = BASELINE - 1;
771 2          END DECBASE;

```

```

/* ----- */

```

```

/* increment limits */
772 1      INCFRONT: PROC; FRONT = FRONT + 1;
774 2          END INCFRONT;
775 1      INCBACK: PROCEDURE; BACK = BACK + 1;
777 2          END INCBACK;

```

```

/* ----- */

```

```

/* decrement limits */
778 1      DECFRONT: PROC; FRONT = FRONT - 1;
780 2          IF MEMORY(FRONT) = LF THEN
781 2              CALL DECBASE;
782 2          END DECFRONT;
783 1      DECBACK: PROC; BACK = BACK - 1;

```

COPYRIGHT ©1982  
 DIGITAL RESEARCH  
 P.O. BOX 579  
 PACIFIC GROVE, CA 93950

SER. # \_\_\_\_\_

```

785 2      END DECBACK;
/* ----- */

/* move current page in memory if move flag
   true otherwise delete it */
786 1  MEM$MOVE: PROC(MOVEFLAG);
787 2      DECLARE (MOVEFLAG,C) BYTE;
/* MOVE IF MOVEFLAG IS TRUE */
788 2      IF DIRECTION = FORWARD THEN
789 2          DO WHILE BACK < LASTC; CALL INCBACK;
791 3          IF MOVEFLAG THEN
792 3              DO;
793 4                  IF (C := MEMORY(BACK)) = LF THEN CALL INCBASE;
795 4                  MEMORY(FRONT) = C; CALL INCFRONT;
797 4                  END;
798 3          END;
      ELSE
799 2          DO WHILE FRONT > FIRST; CALL DECFRONT;
801 3          IF MOVEFLAG THEN
802 3              DO; MEMORY(BACK) = memory(front); CALL DECBACK;
805 4              END;
806 3          END;
807 2      END MEM$MOVE;
/* ----- */

/* force a memory move */
808 1  MOVER: PROC;
809 2      CALL MEM$MOVE(TRUE);
810 2      END MOVER;
/* ----- */

/* reset memory limit pointers, deleting
   characters (used by D command) */
811 1  SETPTRS: PROC;
812 2      CALL MEM$MOVE(FALSE);
813 2      END SETPTRS;
/* ----- */

/* set limits and force a move */
814 1  NOVELINES: PROC;
815 2      CALL SETLIMITS;
816 2      CALL MOVER;
817 2      END NOVELINES;
/* ----- */

/* set front to lower value deleting
   characters (used by S and J commands) */
818 1  setfront: proc(newfront);
819 2      dcl newfront addr;

820 2      do while front > newfront;
821 3          call decfront;

```

COPYRIGHT ©1982  
 DIGITAL RESEARCH  
 P. O. BOX 579  
 PACIFIC GROVE, CA 93950

SER. # \_\_\_\_\_

```

822 3      end;
823 2      end setfront;
/* ----- */

```

```

/* set limits for memory move */
824 1      SETLIMITS: PROC;
825 2          IF DIRECTION = BACKWARD THEN
826 2              DO; LASTC = BACK;
828 3              IF DISTANCE > FRONT THEN
829 3                  FIRST = 1;
                ELSE
830 3                  FIRST = FRONT - DISTANCE;
831 3                  END;
                ELSE
832 2                  DO; FIRST = FRONT;
834 3                  IF DISTANCE >= MAX - BACK THEN
835 3                      LASTC = MAXM;
                ELSE
836 3                      LASTC = BACK + DISTANCE;
837 3                      END;
838 2          END SETLIMITS;
/* ----- */

```

COPYRIGHT ©1982  
 DIGITAL RESEARCH  
 P. O. BOX 579  
 PACIFIC GROVE, CA 93950

SER. # \_\_\_\_\_

```

/* read another line of input */
839 1      READLINE: PROCEDURE;
840 2          DECLARE B BYTE;
                /* READ ANOTHER LINE OF INPUT */
841 2          CTRAN: PROCEDURE(B) BYTE;
842 3              DECLARE B BYTE;
                /* CONDITIONALLY TRANSLATE TO UPPER CASE ON INPUT */
843 3              IF UPPER THEN RETURN UTRAN(B);
845 3              RETURN B;
846 3              END CTRAN;
847 2          DO FOREVER;
848 3              IF FRONT >= BACK THEN GO TO OVERFLOW;
850 3              IF (B := CTRAN(GETSOURCE)) = ENDFILE THEN
851 3                  DO; CALL ZERODIST; RETURN;
854 4                  END;
855 3              MEMORY(FRONT) = B;
856 3              CALL INCFRONT;
857 3              IF B = LF THEN
858 3                  DO; CALL INCBASE;
860 4                  RETURN;
861 4                  END;
862 3              END;
863 2          END READLINE;
/* ----- */

```

```

/* write one line out */
864 1      WRITELINE: PROCEDURE;
865 2          DECLARE B BYTE;
866 2          DO FOREVER;
867 3              IF BACK >= MAXM THEN /* EMPTY */
868 3                  DO; CALL ZERODIST; RETURN;

```



```

871 4      END;
872 3      CALL INCBACK;
873 3      CALL PUTDEST(B:=MEMORY(BACK));
874 3      IF B = LF THEN
875 3          DO; CALL INCBASE;
877 4      RETURN;
878 4      END;
879 3      END;
880 2      END WRITELINE;
/* ----- */

```

```

/* write lines until at least half the
the buffer is empty */
881 1  WRHALF: PROCEDURE;
882 2      CALL SETFF;
883 2      DO WHILE DISTNZERO;
884 3      IF HMAX >= (MAXM - BACK) THEN
885 3          CALL ZERODIST;
886 3      ELSE
887 3          CALL WRITELINE;
888 2      END;
/* ----- */

```

```

/* write lines determined by distance
called from W and E commands */
889 1  WRITEOUT: PROCEDURE;
890 2      DIRECTION = BACKWARD; FIRST = 1; LASTC = BACK;
893 2      CALL MOVER;
894 2      IF DISTZERO THEN CALL WRHALF;
/* DISTANCE = 0 IF CALL WRHALF */
896 2      DO WHILE DISTNZERO;
897 3      CALL WRITELINE;
898 3      END;
899 2      IF BACK < LASTC THEN
900 2          DO; DIRECTION = FORWARD; CALL MOVER;
903 3      END;
904 2      END WRITEOUT;
/* ----- */

```

```

/* clear memory buffer */
905 1  CLEARMEM: PROCEDURE;
906 2      CALL SETFF;
907 2      CALL WRITEOUT;
908 2      END CLEARMEM;
/* ----- */

```

```

/* clear buffers, terminate edit */
909 1  TERMINATE: PROCEDURE;
910 2      CALL CLEARMEM;
911 2      if not newfile then
912 2          DO WHILE (CHAR := GETSOURCE) <> ENDFILE;
913 3          CALL PUTDEST(CHAR);

```

COPYRIGHT ©1982  
 DIGITAL RESEARCH  
 P. O. BOX 579  
 PACIFIC GROVE, CA 93950

SER. # \_\_\_\_\_

914 3  
915 2  
916 2

END;  
CALL FINIS;  
END TERMINATE;

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # \_\_\_\_\_

```
$ eject
```

```
/* ****
```

```
*** COMMAND PRIMITIVES ***
```

```
*****/
```

```
/* insert char into memory buffer */
```

```
917 1  INSERT: PROCEDURE;
918 2      IF FRONT = BACK THEN GO TO OVERFLOW;
920 2      MEMORY(FRONT) = CHAR; CALL INCFRONT;
922 2      IF CHAR = LF THEN CALL INCBASE;
924 2      END INSERT;
/* ----- */
```

```
/* read a character and check for endfile
or CR */
```

```
925 1  SCANNING: PROCEDURE BYTE;
926 2      RETURN NOT ((CHAR := READC) = ENDFILE OR
                      (CHAR = CR AND NOT INSERTING));
927 2      END SCANNING;
/* ----- */
```

```
/* read command buffer and insert characters
into scratch 'til next endfile or CR for
find, next, juxt, or substitute commands
fill at WBE and increment WBE so it
addresses the next empty position of scratch */
```

```
928 1  COLLECT: PROCEDURE;

929 2      SETSCR: PROCEDURE;
930 3          SCRATCH(WBE) = CHAR;
931 3          IF (WBE := WBE + 1) >= SCRSIZE THEN GO TO OVERFLOW;
933 3          END SETSCR;

934 2      DO WHILE SCANNING;
935 3          IF CHAR = CTLL THEN
936 3              DO; CHAR = CR; CALL SETSCR;
939 4              CHAR = LF;
940 4              END;
941 3          IF CHAR = 0 THEN GO TO BADCOM;
943 3          CALL SETSCR;
944 3          END;
945 2      END COLLECT;
/* ----- */
```

```
/* find the string in scratch starting at
PA and ending at PB */
```

COPYRIGHT ©1982  
DIGITAL RESEARCH  
P.O. BOX 579  
PACIFIC GROVE, CA 93950

SER. # \_\_\_\_\_

```

946 1  FIND: PROCEDURE(PA,PB) BYTE;
947 2  DECLARE (PA,PB) BYTE;
948 2  DECLARE J ADDRESS,
      (K, MATCH) BYTE;
949 2  J = BACK ;
950 2  MATCH = FALSE;
951 2  DO WHILE NOT MATCH AND (MAXM > J);
952 3  LASTC, J = J + 1; /* START SCAN AT J */
953 3  K = PA ; /* ATTEMPT STRING MATCH AT K */
954 3  DO WHILE SCRATCH(K) = MEMORY(LASTC) AND
      NOT (MATCH := K = PB);
      /* MATCHED ONE MORE CHARACTER */
955 4  K = K + 1; LASTC = LASTC + 1;
957 4  END;
958 3  END;
959 2  IF MATCH THEN /* MOVE STORAGE */
960 2  DO; LASTC = LASTC - 1; CALL MOVER;
963 3  END;
964 2  RETURN MATCH;
965 2  END FIND;
/* ----- */

```

```

      /* set up the search string for F, N, and
      S commands */
966 1  SETFIND: PROCEDURE;
967 2  WBE = 0; CALL COLLECT; WBP = WBE;
970 2  END SETFIND;
/* ----- */

```

```

      /* check for found string in F and S commands */
971 1  CHKFOUND: PROCEDURE;
972 2  IF NOT FIND(0,WBP) THEN /* NO MATCH */ GO TO OVERCOUNT;
974 2  END CHKFOUND;
/* ----- */

```

```

      /* parse read / xfer lib FCB */
975 1  parse$lib: procedure(fcbadr) byte;
976 2  dcl fcbadr address;
977 2  dcl afcb based fcbadr (33) byte;
978 2  dcl b byte;

979 2  b = parse$fcf(fcbadr);
      /* flag = false if invalid */
980 2  if not flag then do;
982 3  flag = '0';
983 3  goto reset;
984 3  end;
985 2  if afcb(9) = ' ' and not dot$found then
986 2  call move(3,.libfcb(9),fcbadr+9);
987 2  if afcb(1) = ' ' then
988 2  call move(8,.libfcb(1),fcbadr+1);
989 2  return b;
990 2  end parse$lib;
/* ----- */

```

COPYRIGHT ©1982  
DIGITAL RESEARCH  
P.O. BOX 579  
PACIFIC GROVE, CA 93950

SER. # \_\_\_\_\_

```

/* print relative position */
991 1 PRINTREL: PROCEDURE;
992 2     CALL PRINTLINE(BASELINE+RELLINE);
993 2     END PRINTREL;
/* ----- */

```

```

/* type lines command */
994 1 TYPELINES: PROCEDURE;
995 2     DCL I ADDR;
996 2     DCL C BYTE;
997 2     CALL SETLIMITS;
998 2     /* DISABLE THE * PROMPT */
998 2     INSERTING = TRUE;
999 2     IF DIRECTION = FORWARD THEN
1000 2         DO; RELLINE = 0; I = FRONT;
1003 3         END;
1004 2     ELSE
1004 2         I = FIRST;
1005 2     IF (C := MEMORY(I-1)) = LF then do;
1007 3         if COLUMN <> 0 THEN
1008 3             CALL CRLF;
1009 3         end;
1010 2     else
1010 2         relline = relline + 1;
1011 2     DO I = FIRST TO LASTC;
1012 3     IF C = LF THEN
1013 3         DO;
1014 4             CALL PRINTREL;
1015 4             RELLINE = RELLINE + 1;
1016 4             IF BREAK$KEY THEN GO TO OVERCOUNT;
1018 4             END;
1019 3         CALL PRINTC(C:=MEMORY(I));
1020 3         END;
1021 2     END TYPELINES;
/* ----- */

```

COPYRIGHT © 1982  
 DIGITAL RESEARCH  
 P.O. BOX 579  
 PACIFIC GROVE, CA 92050  
 SER. # \_\_\_\_\_

```

/* set distance to lines per page (LPP) */
1022 1 SETLPP: PROCEDURE;
1023 2     DISTANCE = LPP;
1024 2     END SETLPP;
/* ----- */

```

```

/* save distance in TDIST */
1025 1 SAVEDIST: PROCEDURE;
1026 2     TDIST = DISTANCE;
1027 2     END SAVEDIST;
/* ----- */

```

```

/* Restore distance from TDIST */
1028 1 RESTDIST: PROCEDURE;

```

```

1029 2      DISTANCE = TDIST;
1030 2      END RESTDIST;
/* ----- */

```

```

/* page command (move n pages and print) */
1031 1      PAGE: PROCEDURE;
1032 2      DECLARE I BYTE;
1033 2      CALL SAVEDIST;
1034 2      CALL SETLPP;
1035 2      CALL MOVELINES;
1036 2      I = DIRECTION;
1037 2      DIRECTION = FORWARD;
1038 2      CALL SETLPP;
1039 2      CALL TYPELINES;
1040 2      DIRECTION = I;
1041 2      IF LASTC = MAXH OR FIRST = 1 THEN
1042 2          CALL ZERODIST;
      ELSE
1043 2          CALL RESTDIST;
1044 2      END PAGE;
/* ----- */

```

COPYRIGHT ©1982  
 DIGITAL RESEARCH  
 P.O. BOX 579  
 PACIFIC GROVE, CA 93950  
 SER. # \_\_\_\_\_

```

/* wait command (1/2 second time-out) */
1045 1      WAIT: PROCEDURE;
1046 2      DECLARE I BYTE;
1047 2      DO I = 0 TO 19;
1048 3          IF BREAK$KEY THEN GO TO RESET;
1050 3          CALL TIME(250);
1051 3          END;
1052 2      END WAIT;
/* ----- */

```

```

/* set direction to forward */
1053 1      SETFORWARD: PROCEDURE;
1054 2      DIRECTION = FORWARD;
1055 2      DISTANCE = 1;
1056 2      END SETFORWARD;
/* ----- */

```

```

/* append 'til buffer is at least half full */
1057 1      APPHALF: PROCEDURE;
1058 2      CALL SETFF; /* DISTANCE = OFFFHH */
1059 2      DO WHILE DISTNZERO;
1060 3          IF FRONT >= HMAX THEN
1061 3              CALL ZERODIST;
      ELSE
1062 3          CALL READLINE;
1063 3          END;
1064 2      END APPHALF;
/* ----- */

```

```

/* insert CR LF characters */

```

```
1065 1  INSCRLF: PROCEDURE;  
      /* INSERT CR LF CHARACTERS */  
1066 2  CHAR = CR; CALL INSERT;  
1068 2  CHAR = LF; CALL INSERT;  
1070 2  END INSCRLF;  
      /* ----- */  
  
      /* test if invalid delete or  
      backspace at beginning of inserting */  
1071 1  ins$error$chk: procedure;  
1072 2  if (tcolumn = 255) or (front = 1) then  
1073 2  go to reset;  
1074 2  end ins$error$chk;
```

COPYRIGHT ©1982  
DIGITAL RESEARCH  
P.O. BOX 579  
PACIFIC GROVE, CA 93950  
SER. # \_\_\_\_\_

\$ eject

/\* \*\*\*\* \*/

# \*\*\* COMMAND PARSING \*\*\*

\*\*\*\* \*/

/\* test for upper or lower case command  
set translate flag (used to determine  
if following characters should be translated  
to upper case \*/

```
1075 1  TESTCASE: PROCEDURE;
1076 2  DECLARE T BYTE;
1077 2  TRANSLATE = TRUE;
1078 2  T = LOWERCASE(CHAR);
1079 2  CHAR = UTRAN(CHAR);
1080 2  TRANSLATE = UPPER OR NOT T;
1081 2  END TESTCASE;
```

/\* ----- \*/

/\* set translate to false and read next  
character \*/

```
1082 1  READCTAN: PROCEDURE;
1083 2  TRANSLATE = FALSE;
1084 2  CHAR = READC;
1085 2  CALL TESTCASE;
1086 2  END READCTAN;
```

/\* ----- \*/

/\* return true if command is only character  
not in macro or combination on a line \*/

```
1087 1  SINGLECOM: PROCEDURE(C) BOOLEAN;
1088 2  DECLARE C BYTE;
1089 2  RETURN CHAR = C AND COMLEN = 1 AND MP = 0;
1090 2  END SINGLECOM;
```

/\* ----- \*/

/\* return true if command is only character  
not in macro or combination on a line, and  
the operator has responded with a 'Y' to a  
Y/N request \*/

```
1091 1  SINGLERCOM: PROCEDURE(C) BOOLEAN;
1092 2  DECLARE (C,i) BYTE;
1093 2  IF SINGLECOM(C) THEN
1094 2  DO forever;
1095 3  CALL CRLF; CALL PRINTCHAR(C);
1097 3  CALL MON1(9,.(Y/N),WHAT,'$'));
1098 3  i = UCASE(READCHAR); CALL CRLF;
```

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. #



```
1100 3      IF i = 'N' THEN GO TO START;
1102 3      if i = 'Y' then
1103 3          RETURN TRUE;
1104 3      END;
1105 2      RETURN FALSE;
1106 2      END SINGLERCOM;
/* ----- */
```

```
/* return true if char is a digit */
1107 1  DIGIT: PROCEDURE BOOLEAN;
1108 2      RETURN (I := CHAR - '0') <= 9;
1109 2      END DIGIT;
/* ----- */
```

```
/* return with distance = number char =
   next command */
1110 1  NUMBER: PROCEDURE;
1111 2      DISTANCE = 0;
1112 2      DO WHILE DIGIT;
1113 3          DISTANCE = SHL(DISTANCE,3) +
                  SHL(DISTANCE,1) + I;
1114 3          CALL READCTAN;
1115 3          END;
1116 2      END NUMBER;
/* ----- */
```

```
/* set distance to distance relative to
   the current line */
1117 1  RELDISTANCE: PROCEDURE;
1118 2      IF DISTANCE > BASELINE THEN
1119 2          DO; DIRECTION = FORWARD;
1121 3          DISTANCE = DISTANCE - BASELINE;
1122 3          END;
        ELSE
1123 2          DO; DIRECTION = BACKWARD;
1125 3          DISTANCE = BASELINE - DISTANCE;
1126 3          END;
1127 2      END RELDISTANCE;
```

COPYRIGHT ©1982  
DIGITAL RESEARCH  
P. O. BOX 579  
PACIFIC GROVE, CA 93950

SER. # \_\_\_\_\_

```

$ eject

/* **** */

      * * * MAIN PROGRAM * * *

/* **** */

1128 1    plm:          /* entry of HP/M-86 Interface */

      /* INITIALIZE THE SYSTEM */

      ver = version;
1129 1      if low(ver) >= cpm3 then
1130 1          has$bdos3 = true;      /* handles passwords & xfcbs */

      /* * * * * * SET UP MEMORY BUFFER * * * * * */

      /* I/O BUFFER REGION IS 1/8 AVAILABLE MEMORY */
1131 1      NBUF = SHR(MAX := MAXB - .MEMORY,SECTSHF+3) - 1;
      /* NBUF IS NUMBER OF BUFFERS - 1 */
1132 1      BUFLNGTH = SHL(DOUBLE(NBUF+1),SECTSHF+1);
      /* NOW SET MAX AS REMAINDER OF FREE MEMORY */
1133 1      IF BUFLNGTH + 1024 > MAX THEN
1134 1          DO; CALL perror(.'Insufficient memory$');
1136 2          CALL BOOT;
1137 2          END;
      /* REMOVE BUFFER SPACE AND 00 AT END OF MEMORY VECTOR */
1138 1      MAX = MAX - BUFLNGTH - 1;
      /* RESET BUFFER LENGTH FOR I AND O */
1139 1      BUFLNGTH = SHR(BUFLNGTH,1);
1140 1      SBUFFADR = MAXB - BUFLNGTH;
1141 1      DBUFFADR = SBUFFADR - BUFLNGTH;
1142 1      MEMORY(MAX) = 0; /* STOPS MATCH AT END OF BUFFER */
1143 1      MAXH = MAX - 1;
1144 1      HMAX = SHR(MAXH,1);

      /* * * * * * SET UP SOURCE & DESTINATION FILES * * * * * */

1145 1      if fcb(1)=' ' then do;
1147 2          call print(.'Enter Input file: $');
1148 2          call readcom;
1149 2          call crlf;
1150 2          tail = false;
1151 2          end;
1152 1      if not parse$fcb(.SFCB) then      /* parse source fcb */
1153 1          call reboot;

1154 1      if has$bdos3 then do;
1156 2          call read$xfcb(.sfcB);      /* get prot from source */
1157 2          protection = sfcB(ex);      /* password protection mode */
1158 2          sfcB(ex) = 0;
1159 2          if high(ver) = 0 then      /* CP/M-80 */
1160 2              if (lpp:=non2(49,.pb)) = 0 then

```

COPYRIGHT ©1982  
 DIGITAL RESEARCH  
 P.O. BOX 579  
 PACIFIC GROVE, CA 93950

SER. # \_\_\_\_\_

```

1161 2          lpp = 23;          /* get lines per page from SCB */
1162 2          end;
1163 1          call setdest;        /* parse destination file */
1164 1          tail = false;        /* parse fcb from ED command */

```

```
/* SOURCE AND DESTINATION DISKS SET */
```

```
/* IF SOURCE AND DESTINATION DISKS DIFFER, CHECK FOR
AN EXISTING SOURCE FILE ON THE DESTINATION DISK - THERE
COULD BE A FATAL ERROR CONDITION WHICH COULD DESTROY A
FILE IF THE USER HAPPENED TO BE ADDRESSING THE WRONG
DISK */
```

```

1165 1          IF (SDISK <> DDISK) or not onefile THEN
1166 1              IF mon2(15,.dfcb) <> 255 THEN /* try to open */
                  /* SOURCE FILE PRESENT ON DEST DISK */
1167 1              CALL ABORT(,('Output File Exists, Erase It$'));

```

```

1168 1          RESTART:
                  CALL SETUP;
1169 1          MEMORY(0) = LF;
1170 1          FRONT = 1; BACK = MAXM;
1172 1          COLUMN = 0;
1173 1          GO TO START;

```

```
1174 1          OVERCOUNT: FLAG = POUND; GO TO RESET;
```

```
1176 1          BADCOM: FLAG = WHAT; GO TO RESET;
```

```

1178 1          OVERFLOW: /* ARRIVE HERE ON OVERFLOW CONDITION (I,F,S COMMAND) */
                  FLAG = '>'; go to reset;

```

```

1180 1          disk$err:
                  flag = 'F';
1181 1          err$msg = .diskfull;
1182 1          go to reset;

```

```

1183 1          dir$err:
                  flag = 'F';
1184 1          err$msg = .dirfull;

```

```

1185 1          RESET: /* ARRIVE HERE ON ERROR CONDITION */
                  PRINTSUPPRESS = FALSE;
1186 1                  CALL PRINT(, (tab, 'BREAK '$'));
1187 1                  CALL PRINTC(FLAG);
1188 1                  CALL PRINTN(, (' AT '$'));
1189 1                  if char = CR or char = LF then
1190 1                      call printn(, ('END OF LINE$'));
                  else
1191 1                      CALL PRINTC(CHAR);
1192 1                  if err$msg <> 0 then do;
1194 2                      call perror(err$msg);
1195 2                      err$msg = 0;
1196 2                      end;
1197 1                  CALL CRLF;

```

COPYRIGHT © 1982  
 DIGITAL RESEARCH  
 P. O. BOX 579  
 PACIFIC GROVE, CA 93950

SER. # \_\_\_\_\_

1198 1 START:  
1199 1 READBUFF = TRUE;  
HP = 0;

COPYRIGHT ©1982  
DIGITAL RESEARCH  
P.O. BOX 579  
PACIFIC GROVE, CA 93950

SER. #

\$ eject

```

1200 1      DO FOREVER; /* OR UNTIL THE POWER IS TURNED OFF */

/* *****
SIMPLE COMMANDS (CANNOT BE PRECEDED BY DIRECTION/DISTANCE):
    E      END THE EDIT NORMALLY
    H      MOVE TO HEAD OF EDITED FILE
    I      INSERT CHARACTERS
    O      RETURN TO THE ORIGINAL FILE
    R      READ FROM LIBRARY FILE
    Q      QUIT EDIT WITHOUT CHANGES TO ORIGINAL FILE
***** */

1201 2      INSERTING = FALSE;
1202 2      CALL READCTAN;
1203 2      FLAG = 'E';
1204 2      MI = CBP; /* SAVE STARTING ADDRESS FOR <CR> COMMAND */
1205 2      IF SINGLECON('E') THEN
1206 2          DO; CALL TERMINATE;
1208 3          CALL REBOOT;
1209 3          END;

1210 2      ELSE IF SINGLECON('H') THEN /* GO TO TOP */
1211 2          DO; CALL TERMINATE;
1213 3          newfile = false;
1214 3          if onefile then do;
1216 4              /* PING - PONG DISKS */
1217 4              CHAR = DDISK;
1218 4              DDISK = SDISK;
1219 4              SDISK = CHAR;
1220 3          end;
1221 4          else do;
1222 4              call settype(.dfcb,.dtype);
1223 4              call move(16,.dfcb,.sfcb); /* source = destination */
1224 4              onefile = true;
1225 3          end;
1226 3          GO TO RESTART;
1227 2      ELSE IF CHAR = 'I' THEN /* INSERT CHARACTERS */
1228 2          DO;
1229 3          IF (INSERTING := (CBP = COMLEN) AND (MP = 0)) THEN do;
1231 4              tcolumn = 255; /* tested in ins$error$chk routine */
1232 4              distance = 0;
1233 4              direction = backward;
1234 4              if memory(front-1) = LF then
1235 4                  call printbase;
1236 4              else
1237 4                  call typelines;
1238 3          DO WHILE SCANNING;
1239 4              DO WHILE CHAR <> 0;
1240 5              IF CHAR=CTLU OR CHAR=CTLX OR CHAR=CTLR THEN
                  /* LINE DELETE OR RETYPE */

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # \_\_\_\_\_

```

1241 5      DO;
          /* ELIMINATE OR REPEAT THE LINE */
1242 6      IF CHAR = CTLR THEN
1243 6          DO; CALL CRLF;
1244 7          CALL TYPELINES;
1245 7          END;
          ELSE
          /* LINE DELETE */
1247 6          DO; CALL SETLIMITS; CALL SETPTRS;
1250 7          IF CHAR = CTLU THEN
1251 7              DO; CALL CRLF; CALL PRINTNMBASE;
1254 8              END;
          ELSE
          /* MUST BE CTLX */
1255 7          DO WHILE COLUMN > SCOLUMN;
1256 8              CALL BACKSPACE;
1257 8              END;
1258 7          END;
1259 6      END;
1260 5      ELSE IF CHAR = CTLH THEN
1261 5          DO;
1262 6          call ins$error$chk;
1263 6          IF (TCOLUMN := COLUMN) > 0 THEN
1264 6              CALL PRINTMAC(' '); /* RESTORE AFT BACKSP */
1265 6          call decfront;
1266 6          if tcolum > scolum then
1267 6              DO; /* CHARACTER CAN BE ELIMINATED */
1268 7              PRINTSUPPRESS = TRUE;
              /* BACKSPACE CHARACTER ACCEPTED */
              COLUMN = 0;
              CALL TYPELINES;
              PRINTSUPPRESS = FALSE;
              /* COLUMN POSITION NOW RESET */
              IF (QCOLUMN := COLUMN) < SCOLUMN THEN
                  QCOLUMN = SCOLUMN;
              COLUMN = TCOLUMN; /* ORIGINAL VALUE */
              DO WHILE COLUMN > QCOLUMN;
                  CALL BACKSPACE;
              END;
              END;
1278 7          END;
          else
1279 6          do;
1280 7          if memory(front-1) = CR then
1281 7              call decfront;
1282 7          call crlf;
1283 7          call typelines;
1284 7          end;
1285 6          CHAR = 0;
1286 6          END;
1287 5      ELSE IF CHAR = RUBOUT THEN
1288 5          DO; call ins$error$chk;
1289 6          CALL DECFRONT; CALL PRINTC(CHAR:=MEMORY(FRONT));
1290 6          CHAR = 0;
1291 6          END;
1292 6          else if char = LF and memory(front-1) <> CR then
1293 6          do;
1294 5          call printc(CR);

```

COPYRIGHT ©1982  
 DIGITAL RESEARCH  
 P. O. BOX 579  
 PACIFIC GROVE, CA 93950

SER. # \_\_\_\_\_

```

1297 6      call inscrlf;
1298 6      end;
      ELSE
      /* NOT A SPECIAL CASE */
1299 5      DO;
1300 6      IF NOT GRAPHIC(CHAR) THEN
1301 6          DO;
1302 7          CALL PRINTNMAC('^');
1303 7          CALL PRINTNMAC(CHAR + 'Q');
1304 7          end;
      /* COLUMN COUNT GOES UP IF GRAPHIC */
      /* COMPUTE OUTPUT COLUMN POSITION */
1305 6      if char = CTLL and not inserting then
1306 6          call inscrlf;
1307 6      else do;
1308 7          IF MP = 0 THEN
1309 7              DO;
1310 8              IF CHAR >= ' ' THEN
1311 8                  COLUMN = COLUMN + 1;
1312 8              ELSE IF CHAR = TAB THEN
1313 8                  COLUMN = COLUMN + (8 - (COLUMN AND 111B));
              END;
1315 7          CALL INSERT;
1316 7          END;
1317 6      end;
1318 5      IF CHAR = LF THEN CALL PRINTNMBASE;
1320 5      IF CHAR = CR THEN
1321 5          CALL PRINTNMAC(CHAR:=LF);
      ELSE
1322 5          CHAR = 0;
1323 5          tcolumn = 0;
1324 5          END; /* of while char <> 0 */
1325 4      END; /* of while scanning */
1326 3      IF CHAR <> ENDFILE THEN do; /* MUST HAVE STOPPED ON CR */
1328 4          CALL INSCRLF;
1329 4          column = 0;
1330 4          end;
1331 3      IF INSERTING AND LINESET THEN CALL CRLF;
1333 3      END;

1334 2      ELSE IF SINGLERCOM('O') THEN /* FORGET THIS EDIT */
1335 2          do;
1336 3          call close(.sfcb);
1337 3          GO TO RESTART;
1338 3          end;

1339 2      ELSE IF CHAR = 'R' THEN
1340 2          DO; DECLARE I BYTE;
      /* READ FROM LIB FILE */
1342 3          CALL SETRDMA;
1343 3          IF (FLAG := parse$lib(.rfcb)) THEN
1344 3              reading = false;
1345 3          if not reading then do;
1347 4              if not flag then
      /* READ FROM XFER FILE */

```

COPYRIGHT © 1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # \_\_\_\_\_

```

1348 4          CALL MOVE(12,.XFCB,.RFCB);
1349 4          RFCB(12), RFCB(32) = 0; /* zero extent, next record */
1350 4          rbp = sectsize;
1351 4          CALL open(.RFCB);
1352 4          reading = true;
1353 4          end;

1354 3          DO WHILE (CHAR != READFILE) <> ENDFILE;
1355 4          CALL INSERT;
1356 4          END;
1357 3          reading = false;
1358 3          call close (.rfcb);
1359 3          END;

```

```

1360 2          ELSE IF SINGLERCOM('Q') THEN
1361 2              DO;
1362 3              CALL DELETE$FILE(.DFCB);
1363 3              IF NEWFILE OR NOT ONEFILE THEN DO;
1364 4                  call settype(.dfcb,.dtype);
1365 4                  call delete$FILE(.dfcb);
1366 4                  end;
1367 3              CALL REBOOT;
1368 3              END;
1369 3

```

COPYRIGHT ©1982  
 DIGITAL RESEARCH  
 P.O. BOX 579  
 PACIFIC GROVE, CA 93950

SER. # \_\_\_\_\_

ELSE

```

/* MAY BE A COMMAND WHICH HAS AN OPTIONAL DIRECTION AND DISTANCE */
1370 2          DO; /* SCAN A SIGNED INTEGER VALUE (IF ANY) */
1371 3          DCL I BYTE;

1372 3          CALL SETFORWARD;

1373 3          IF CHAR = '-' THEN
1374 3              DO; CALL READCTAN; DIRECTION = BACKWARD;
1375 4              END;

1376 3          IF CHAR = POUND THEN
1377 3              DO; CALL SETFF; CALL READCTAN;
1378 4              END;

1379 3          ELSE IF DIGIT THEN
1380 3              DO; CALL NUMBER;
1381 4              /* MAY BE ABSOLUTE LINE REFERENCE */
1382 4              IF CHAR = ':' THEN
1383 4                  DO; CHAR = 'L';
1384 5                  CALL RELDISTANCE;
1385 5                  END;
1386 4              END;

1387 3          ELSE IF CHAR = ':' THEN /* LEADING COLON */
1388 3              DO; CALL READCTAN; /* CLEAR THE COLON */
1389 4              CALL NUMBER;
1390 4              CALL RELDISTANCE;
1391 4              IF DIRECTION = FORWARD THEN
1392 4                  DISTANCE = DISTANCE + 1;
1393 4              END;
1394 3

```



COPYRIGHT ©1982  
DIGITAL RESEARCH  
P.O. BOX 579  
PACIFIC GROVE, CA 93950

SER. # \_\_\_\_\_

\$ eject

```

1401 3      IF DISTZERO THEN
           DIRECTION = BACKWARD;
           /* DIRECTION AND DISTANCE ARE NOW SET */

```

```

/* *****
MAY BE A COMMAND WHICH HAS DIRECTION AND DISTANCE SPECIFIED:

```

```

      B      BEGINNING/BOTTOM OF BUFFER
      C      MOVE CHARACTER POSITIONS
      D      DELETE CHARACTERS
      K      KILL LINES
      L      MOVE LINE POSITION
      P      PAGE UP OR DOWN (LPP LINES AND PRINT)
      T      TYPE LINES
      U      UPPER CASE TRANSLATE
      V      VERIFY LINE NUMBERS
      <CR>   MOVE UP OR DOWN LINES AND PRINT LINE

```

```

***** */

```

```

1402 3      IF CHAR = 'B' THEN
1403 3          DO; DIRECTION = 1 - DIRECTION;
1405 4          FIRST = 1; LASTC = MAXM; CALL MOVER;
1408 4          END;

```

```

1409 3      ELSE IF CHAR = 'C' THEN
1410 3          DO; CALL SETCLIMITS; CALL MOVER;
1413 4          END;

```

```

1414 3      ELSE IF CHAR = 'D' THEN
1415 3          DO; CALL SETCLIMITS;
1417 4          CALL SETPTRS; /* SETS BACK/FRONT */
1418 4          END;

```

```

1419 3      ELSE IF CHAR = 'K' THEN
1420 3          DO; CALL SETLIMITS;
1422 4          CALL SETPTRS;
1423 4          END;

```

```

1424 3      ELSE IF CHAR = 'L' THEN
1425 3          CALL MOVELINES;

```

```

1426 3      ELSE IF CHAR = 'P' THEN /* PAGE MODE PRINT */
1427 3          DO;
1428 4          IF DISTZERO THEN
1429 4              DO; DIRECTION = FORWARD;
1431 5              CALL SETLPP; CALL TYPelines;
1433 5              END;
           ELSE
1434 4              DO WHILE DISTNZERO; CALL PAGE;

```

COPYRIGHT ©1982  
 DIGITAL RESEARCH  
 P.O. BOX 579  
 PACIFIC GROVE, CA 93950

SER. # \_\_\_\_\_

```

1436 5      CALL WAIT;
1437 5      END;
1438 4      END;

```

```

1439 3      ELSE IF CHAR = 'T' THEN
1440 3      CALL TYPELINES;

```

```

1441 3      ELSE IF CHAR = 'U' THEN
1442 3      UPPER = DIRECTION = FORWARD;

```

```

1443 3      ELSE IF CHAR = 'V' THEN
1444 3      DO; /* OV DISPLAYS BUFFER STATE */
1445 4      IF DISTZERO THEN
1446 4      DO; CALL PRINTVALUE(BACK-FRONT);
1448 5      CALL PRINTC('/');
1449 5      CALL PRINTVALUE(MAXM);
1450 5      CALL CRLF;
1451 5      END;
1452 4      ELSE if (LINESET := DIRECTION = FORWARD) then
1453 4      scolumn = 8;
1454 4      else
1455 4      scolumn = 0;
1456 4      END;

```

```

1456 3      ELSE IF CHAR = CR THEN /* MAY BE MOVE/TYPE COMMAND */
1457 3      DO;
1458 4      IF MI = 1 AND MP = 0 THEN /* FIRST COMMAND */
1459 4      DO; CALL MOVELINES; CALL SETFORWARD; CALL TYPELINES;
1463 5      END;
1464 4      END;

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # \_\_\_\_\_

\$ eject

```

1465 3      ELSE IF DIRECTION = FORWARD OR DISTZERO THEN
1466 3      DO;

```

```

/* *****
COMMANDS WHICH ALLOW ONLY A PRECEDING NUMBER:

```

```

A      APPEND LINES
F      FIND NTH OCCURRENCE
M      APPLY MACRO
N      SAME AS F WITH AUTOSCAN THROUGH FILE
S      PERFORM N SUBSTITUTIONS
W      WRITE LINES TO OUTPUT FILE
X      TRANSFER (XFER) LINES TO TEMP FILE
Z      SLEEP

```

```

***** */

```

```

1467 4      IF CHAR = 'A' THEN
1468 4      DO; DIRECTION = FORWARD;
1470 5      FIRST = FRONT; LASTC = MAXM; CALL MOVER;
          /* ALL STORAGE FORWARD */
1473 5      IF DISTZERO THEN CALL APPHALF;
          /* DISTANCE = 0 IF APPHALF CALLED */
1475 5      DO WHILE DISTNZERO;
1476 6      CALL READLINE;
1477 6      END;
1478 5      DIRECTION = BACKWARD; CALL MOVER;
          /* POINTERS REPOSITIONED */
1480 5      END;

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # \_\_\_\_\_

```

1481 4      ELSE IF CHAR = 'F' THEN
1482 4      DO; CALL SETFIND; /* SEARCH STRING SCANNED
          AND SETUP BETWEEN 0 AND WBP-1 IN SCRATCH */
1484 5      DO WHILE DISTNZERO; CALL CHKFOUND;
1486 6      END;
1487 5      END;

```

```

1488 4      ELSE IF CHAR = 'J' THEN /* JUXTAPOSITION OPERATION */
1489 4      DO; DECLARE T ADDRESS;
1491 5      CALL SETFIND; CALL COLLECT;
1493 5      WBJ = WBE; CALL COLLECT;
          /* SEARCH FOR STRING 0 - WBP-1, INSERT STRING WBP TO WBJ-1,
          AND THEN DELETE UP TO STRING WBJ TO WBE-1 */
1495 5      DO WHILE DISTNZERO; CALL CHKFOUND;
1497 6      /* INSERT STRING */ MI = WBP - 1;
1498 6      DO WHILE (MI := MI + 1) < WBJ;
1499 7      CHAR = SCRATCH(MI); CALL INSERT;
1501 7      END;
1502 6      T = FRONT; /* SAVE POSITION FOR DELETE */
1503 6      IF NOT FIND(WBJ,WBE) THEN GO TO OVERCOUNT;
          /* STRING FOUND, SO MOVE IT BACK */
1505 6      FIRST = FRONT - (WBE - WBJ);
1506 6      DIRECTION = BACKWARD; CALL MOVER;

```

```

/* NOW REMOVE THE INTERMEDIATE STRING */
1508 6      call setfront(t);
1509 6      END;
1510 5      END;

1511 4      ELSE IF CHAR = 'M' AND MP = 0 THEN /* MACRO DEFINITION */
1512 4          DO; XP = 255;
1514 5          IF DISTANCE = 1 THEN CALL ZERODIST;
1516 5          DO WHILE (MACRO(XP := XP + 1) := READC) <> CR;
1517 6          END;
1518 5          MP = XP; XP = 0; MT = DISTANCE;
1521 5          END;

1522 4      ELSE IF CHAR = 'N' THEN
1523 4          DO; /* SEARCH FOR STRING WITH AUTOSCAN */
1524 5          CALL SETFIND; /* SEARCH STRING SCANNED */
1525 5          DO WHILE DISTNZERO;
1526 6              /* FIND ANOTHER OCCURRENCE OF STRING */
1527 7              DO WHILE NOT FIND(0,WBP); /* NOT IN BUFFER */
1529 7              IF BREAK$KEY THEN GO TO RESET;
1531 7              CALL SAVEDIST; CALL CLEARMEN;
1532 7              /* MEMORY BUFFER WRITTEN */
1533 7              CALL APPHALF;
1534 7              DIRECTION = BACKWARD; FIRST = 1; CALL MOVER;
1535 7              CALL RESTDIST; DIRECTION = FORWARD;
1536 7              /* MAY BE END OF FILE */
1537 7              IF BACK >= MAXM THEN GO TO OVERCOUNT;
1539 7              END;
1540 6          END;
1541 5      END;

1542 4      ELSE IF CHAR = 'S' THEN /* SUBSTITUTE COMMAND */
1543 4          DO; CALL SETFIND;
1544 5          CALL COLLECT;
1545 5          /* FIND STRING FROM 0 TO WBP-1, SUBSTITUTE STRING
1546 5          BETWEEN WBP AND WBE-1 IN SCRATCH */
1547 6          DO WHILE DISTNZERO;
1548 6          CALL CHKFOUND;
1549 6          /* FRONT AND BACK NOW POSITIONED AT FOUND
1550 6          STRING - REPLACE IT */
1551 6          call setfront(FRONT - (MI := WBP)); /* BACKED UP */
1552 6          DO WHILE MI < WBE;
1553 6          CHAR = SCRATCH(MI);
1554 6          MI = MI + 1; CALL INSERT;
1555 6          END;
1556 5          END;

1556 4      ELSE IF CHAR = 'W' THEN
1557 4          CALL WRITEOUT;

1558 4      ELSE IF CHAR = 'X' THEN /* TRANSFER LINES */

```

COPYRIGHT © 1982  
 DIGITAL RESEARCH  
 P. O. BOX 579  
 PACIFIC GROVE, CA 93950

SER. # \_\_\_\_\_

```

1559 4      DO;
1560 5      flag = parse$lib(.rfcb);
1561 5      xbp = 0;
1562 5      IF DISTZERO THEN
1563 5          DO;          /* delete the file */
1564 6          xferon = false;
1565 6          CALL DELETE(.rfcb);
1566 6          if dcnt = 255 then
1567 6              call perror(.not$found);
1568 6          END;
      ELSE
1569 5      do;          /* transfer lines */
1570 6      declare i address;

1571 6      if xferon and compare$xfer then
1572 6          call append$xfer;
      else
1573 6          DO;
1574 7          XFERON = TRUE;
1575 7          call move(12,.rfcb,.xpcb);
1576 7          xpcbext, xpcbrec, xpcbe, xpcbr = 0;
1577 7          CALL MAKE$file(.XFCB);
1578 7          IF DCNT = 255 THEN
1579 7              goto dir$err;
1580 7          END;
1581 6          CALL SETLIMITS;
1582 6          DO I = FIRST TO LASTC;
1583 7          CALL PUTXFER(MEMORY(I));
1584 7          END;
1585 6          call close$xfer;
1586 6          END;
1587 5      END;

1588 4      ELSE IF CHAR = 'Z' THEN /* SLEEP */
1589 4      DO;
1590 5      IF DISTZERO THEN
1591 5          DO; IF READCHAR = ENDFILE THEN GO TO RESET;
1594 6          END;
1595 5          DO WHILE DISTNZERO; CALL WAIT;
1597 6          END;
1598 5      END;
1599 4      ELSE IF CHAR <> 0 THEN /* NOT BREAK LEFT OVER FROM STOP */
/* DIRECTION FORWARD, BUT NOT ONE OF THE ABOVE */
1600 4      GO TO BADCOM;

      END;
      ELSE /* DIRECTION NOT FORWARD */
1602 3      GO TO BADCOM;
1603 3      END;
1604 2      END;
1605 1      END;

```

COPYRIGHT © 1982  
 DIGITAL RESEARCH  
 P.O. BOX 879  
 PACIFIC GROVE, CA 93950

SER. # \_\_\_\_\_

CODE AREA SIZE = 1FF1H 8177D  
VARIABLE AREA SIZE = 0360H 864D  
MAXIMUM STACK SIZE = 001CH 28D  
2655 LINES READ  
0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION

COPYRIGHT ©1982  
DIGITAL RESEARCH  
P. O. BOX 579  
PACIFIC GROVE, CA 93950

SER. # \_\_\_\_\_

|                   |                    |                 |                  |                 |
|-------------------|--------------------|-----------------|------------------|-----------------|
| 0000 ED           |                    |                 |                  |                 |
| 0000 MCD80A       |                    |                 |                  |                 |
| 0004 BDISK        | 0080 BUFF          | 0080 BUFFA      | 0050 CHDRV       | 0000 CPU        |
| 007C CR           | 006D DOLLA         | 005C FCB        | 006C FCB16       | 005C FCBA       |
| 005C IFCB         | 005C IFCBA         | 0003 IOBYTE     | 0053 LENO        | 0056 LEN1       |
| 0006 MAXB         | 0006 MEMSIZ        | 0005 MON1       | 0005 MON2        | 0005 MON2A      |
| 0005 MON3         | 0000 OFFSET        | 006E PARMA      | 0051 PASSO       | 0054 PASS1      |
| 007F RO           | 007D RR            | 007D RRECA      | 007F RRECO       | 005C SFCB       |
| 0080 TBUFF        |                    |                 |                  |                 |
| 0000 ED           |                    |                 |                  |                 |
| 25A9 MEMORY       | 02C1 PLM           | 0CB5 BOOT       | 0180 DATE        | 2249 MAX        |
| 224B MAXM         | 224D HMAX          | 224F I          | 2250 RFCB        | 2271 RBP        |
| 2272 XFCB         | 227E XFCBE         | 2292 XFCBR      | 2293 XFCBEXT     | 2294 XFCBREC    |
| 2295 XBUFF        | 2315 XBP           | 2316 NBUF       | 2317 BUFFLENGTH  | 2319 SBUFFADR   |
| 231B PASSWORD     | 232B DFCB          | 232B DDISK      | 234C DBUFFADR    | 234E NSOURCE    |
| 2350 NDEST        | 2352 TMPFCB        | 2373 NEWFILE    | 2374 ONEFILE     | 2375 XFERON     |
| 2376 READING      | 2377 PRINTSUPPRESS |                 | 2378 SYS         | 2379 PROTECTION |
| 237A INSERTING    | 237B READBUFF      | 237C TRANSLATE  | 237D UPPER       | 237E LINESET    |
| 237F HASBDO3      | 2380 TAIL          | 2381 DOTFOUND   | 2382 DTYPE       | 2385 LIBFCB     |
| 2391 TEMPFL       | 2394 BACKUP        | 2397 ERRORCODE  | 2399 COLUMN      | 239A SCOLUMN    |
| 239B TCOLUMN      | 239C QCOLUMN       | 239D DCNT       | 239E MAXLEN      | 239F COMLEN     |
| 23A0 COMBUFF      | 2420 CBP           | 2421 BASELINE   | 2423 RELLINE     | 2425 MACRO      |
| 24A5 SCRATCH      | 2509 WBP           | 250A WBE        | 250B WBJ         | 250C FLAG       |
| 250D HP           | 250E HI            | 250F XP         | 2510 MT          | 04A0 START      |
| 03E7 RESTART      | 0403 OVERCOUNT     | 0419 OVERFLOW   | 0424 DISKERR     | 0435 DIRERR     |
| 0440 RESET        | 040E BADCOM        | 2512 NCMD       | 2513 DISTANCE    | 2515 TDIST      |
| 2517 DIRECTION    | 2518 CHAR          | 2519 FRONT      | 251B BACK        | 251D FIRST      |
| 251F LASTC        | 2521 LPP           | 0184 PB         | 2522 VER         | 2524 ERRMSG     |
| 0186 INVALID      | 0197 DIRFULL       | 01A6 DISKFULL   | 01B0 PASSWORDERR |                 |
| 01C2 NOTFOUND     | 01D1 NOTAVAIL      | 0CBE READCHAR   | 0CC7 CONIN       | 0CDO PRINTCHAR  |
| 2526 CHAR         | 0CEB TTYCHAR       | 2527 CHAR       | 0D0D BACKSPACE   | 0D2B PRINTABS   |
| 2528 CHAR         | 2529 I             | 252A J          | 0D6F GRAPHIC     | 252B C          |
| 0D9F PRINTC       | 252C C             | 0DC3 CRLF       | 0DCE PRINTM      | 252D A          |
| 0DDE PRINT        | 252F A             | 0DF0 PERROR     | 2531 A           | 0E08 READ       |
| 2533 A            | 0E18 OPEN          | 2535 FCB        | 0E3B OPENFILE    | 2537 FCB        |
| 0E4B CLOSE        | 2539 FCB           | 0E5E DELETE     | 253B FCB         | 0E71 DISKREAD   |
| 253D FCB          | 0E81 DISKWRITE     | 253F FCB        | 0E91 RENAME      | 2541 FCB        |
| 0EA1 READCOM      | 0EAD BREAKKEY      | 0ECC CSELECT    | 0ED5 SETDMA      | 2543 A          |
| 0EE5 SETATTRIBUTE |                    | 2545 FCB        | 0EF5 MOVE        | 2547 C          |
| 2548 S            | 254A D             | 0F2C WRITEXFCB  | 254C FCB         | 0F53 READXFCB   |
| 254E FCB          | 0F63 RETURNERRORS  |                 | 2550 MODE        | 0F73 REBOOT     |
| 0F84 VERSION      | 0F8D ABORT         | 2551 A          | 0F9F FERR        |                 |
| 0FAC SETPASSWORD  |                    | 0FBA DELETEFILE | 2553 AFCB        | 0FCC RENAMEFILE |
| 2555 AFCB         | 0FEA MAKEFILE      | 2557 AFCB       | 1008 FILL        | 2559 S          |
| 255B F            | 255C C             | 1033 SETTYPE    | 255D AFCB        | 255F A          |
| 1051 SETXDMA      | 1058 FILLSOURCE    | 2561 I          | 1089 ZN          | 10C0 GETSOURCE  |
| 2562 B            | 10F5 ERASEBAK      | 112C WRITEDEST  | 2563 I           | 2564 N          |
| 2565 SAVENDEST    | 115D RETRY         | 11CD PUTDEST    | 2566 B           | 11F4 PUTXFER    |
| 2567 C            | 1200 RETRY         | 123E CLOSEXFER  | 2568 I           |                 |
| 1260 COMPAREXFER  |                    | 2569 I          | 1293 APPENDXFER  | 12DF FINIS      |
| 135D MOVEUP       | 256A AFCB          | 1377 PRINTNMAC  | 256C CHAR        | 138C LOWERCASE  |
| 256D C            | 13A4 UCASE         | 256E C          | 13BD GETPASSWD   | 256F I          |
| 2570 C            | 13C6 RETRY         | 13DF NXTCHR     | 1449 EXIT        | 1450 UTRAN      |
| 2571 C            | 1474 PRINTVALUE    | 2572 V          | 2574 D           | 2575 ZERO       |
| 2576 K            | 14DF PRINTLINE     | 2578 V          | 1514 PRINTBASE   |                 |
| 151D PRINTNMBASE  |                    | 152A GETCMD     | 154C READC       | 162E GETUC      |
| 164D PARSEFCB     | 257A FCBADR        | 257C I          | 257D DELIMITER   | 257E PFLAG      |
| 17A1 PUTC         | 17B9 DELIM         | 01E4 DEL        | 1793 ERR         | 17F3 SETDEST    |
| 257F I            | 1852 SETDMA        | 1859 READFILE   | 188B SETUP       | 19BB SETFF      |
| 19C2 DISTZERO     | 19CF ZERODIST      | 19D6 DISTNZERO  | 19EA SETLIMITS   | 2580 I          |
| 2582 K            | 2584 L             | 2586 N          | 2588 MIDDLE      | 2589 LOOPING    |
| 1AD3 INCBASE      | 1ADB DECBASE       | 1AE3 INCFRONT   | 1AEB INCBACK     | 1AF3 DECFRONT   |
| 1B08 DECBACK      | 1B10 MEMMOVE       | 258A MOVEFLAG   | 258B C           | 1885 MOVER      |

COPYRIGHT ©1982  
DIGITAL RESEARCH  
P.O. BOX 579  
PACIFIC GROVE, CA 93950

SER. #



|                  |                 |                  |               |                 |
|------------------|-----------------|------------------|---------------|-----------------|
| 188B SETPTRS     | 1891 MOVELINES  | 1898 SETFRONT    | 258C NEWFRONT | 18B2 SETCLIMITS |
| 1C11 READLINE    | 258E B          | 1C51 CTRAN       | 258F B        | 1C68 WRITELINE  |
| 2590 B           | 1C9A WRHALF     | 1CC3 WRITEOUT    | 1D03 CLEARMEH | 1D0A TERMINATE  |
| 1D2D INSERT      | 1D57 SCANNING   | 1D78 COLLECT     | 1DA6 SETSCR   | 1DC3 FIND       |
| 2591 PA          | 2592 PB         | 2593 J           | 2595 K        | 2596 MATCH      |
| 1E4D SETFIND     | 1E5C CHKFOUND   | 1E6D PARSELIB    | 2597 FCBADR   | 2599 B          |
| 1ED3 PRINTREL    | 1EE1 TYPELINES  | 259A I           | 259C C        | 1F77 SETLPP     |
| 1F80 SAVEDIST    | 1F87 RESTDIST   | 1F8E PAGE        | 259D I        | 1FD9 WAIT       |
| 259E I           | 1FFE SETFORWARD | 200A APPHALF     | 202D INSCRLF  |                 |
| 203E INSERRORCHK |                 | 205E TESTCASE    | 259F T        | 2082 READCTAN   |
| 2091 SINGLECOM   | 25A0 C          | 2088 SINGLERCOM  | 25A1 C        | 25A2 I          |
| 2102 DIGIT       | 2111 NUMBER     | 213D RELDISTANCE |               | 25A3 I          |
| 25A4 I           | 25A5 T          | 25A7 I           |               |                 |

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # \_\_\_\_\_

0000 ED#  
0000 MCD80A#  
0000 ED#

|         |         |          |          |          |
|---------|---------|----------|----------|----------|
| OCB5 20 | OCB5 21 | OCBD 22  | OCBE 45  | OCBE 46  |
| OCC7 47 | OCC7 48 | OCC7 49  | OCDO 50  | OCDO 51  |
| OCDA 53 | OCDB 54 | OCDC 55  | OCE7 56  | OCEB 57  |
| OCEC 59 | OCF4 60 | OCF8 61  | OD00 62  | OD05 63  |
| OD0C 64 | OD0D 65 | OD0D 66  | OD15 67  | OD16 68  |
| OD1B 69 | OD20 70 | OD25 71  | OD2A 72  | OD2B 73  |
| OD2F 75 | OD47 76 | OD4C 77  | OD51 78  | OD60 79  |
| OD67 80 | OD6E 81 | OD6F 82  | OD73 84  | OD7B 85  |
| OD7E 86 | OD9F 87 | OD9F 88  | ODA3 90  | ODAE 92  |
| ODB3 93 | ODBB 94 | ODBB 95  | ODC2 96  | ODC3 97  |
| ODC3 98 | ODCB 99 | ODCD 100 | ODCE 101 | ODDA 103 |

|          |          |          |          |          |
|----------|----------|----------|----------|----------|
| ODDD 104 | ODDE 105 | ODE4 107 | ODE7 108 | ODEF 109 |
| ODF0 110 | ODF6 112 | ODFC 113 | OE04 114 | OE07 115 |
| OE08 116 | OE0E 118 | OE17 119 | OE18 120 | OE1E 122 |
| OE2C 124 | OE31 125 | OE37 126 | OE3A 127 | OE3A 128 |
| OE3B 129 | OE41 131 | OE4B 132 | OE4B 133 | OE51 135 |
| OE5D 136 | OE5E 137 | OE64 139 | OE70 140 | OE71 141 |
| OE77 143 | OE81 144 | OE81 145 | OE87 147 | OE91 148 |
| OE91 149 | OE97 151 | OEAO 152 | OEAI 153 | OEAI 154 |
| OEAA 155 | OEAC 156 | OEAD 157 | OEAD 158 | OE89 160 |
| OECA 161 | OEC9 162 | OEC9 163 | OECB 164 | OECB 165 |
| OECB 166 | OED5 167 | OED5 168 | OEDB 170 | OEE4 171 |
| OEE5 172 | OEEB 174 | OEFA 175 | OEFA 176 | OF04 179 |
| OF10 180 | OF1A 181 | OF21 182 | OF28 183 | OF2B 184 |
| OF2C 185 | OF32 187 | OF3E 188 | OF4C 189 | OF52 190 |
| OF53 191 | OF59 193 | OF62 194 | OF63 195 | OF67 197 |
| OF72 198 | OF73 199 | OF73 200 | OF7A 201 | OF80 202 |
| OF83 203 | OF84 204 | OF84 205 | OF8D 206 | OF8D 207 |
| OF93 209 | OF9B 210 | OF9E 211 | OF9F 212 | OF9F 213 |
| OFAS 214 | OFAB 215 | OFAC 216 | OFAC 217 | OFB3 218 |
| OFB9 219 | OFBA 220 | OFCA 222 | OFCA 223 | OFCA 224 |
| OFCC 225 | OFD2 227 | OFDE 228 | OFD1 229 | OFD9 230 |
| OFDA 231 | OFF0 233 | OFFB 234 | OFFB 235 | 1007 236 |
| 1008 237 | 1015 239 | 1021 240 | 1028 241 | 102F 242 |
| 1032 243 | 1033 244 | 103D 246 | 1050 247 | 1051 248 |
| 1051 249 | 1057 250 | 1058 251 | 10B9 253 | 10B9 254 |
| 10BF 255 | 1058 256 | 105B 257 | 106A 258 | 1077 259 |
| 1085 261 | 108E 262 | 1091 263 | 109B 264 | 10A1 265 |
| 10A4 266 | 10AE 267 | 10B5 268 | 10B8 269 | 10C0 270 |
| 10C0 272 | 10C7 273 | 10CA 274 | 10D6 275 | 10D9 276 |
| 10EA 277 | 10F1 278 | 10F5 279 | 10F5 280 | 10F5 281 |
| 10FC 282 | 1103 284 | 110F 285 | 1118 286 | 111E 287 |
| 1126 288 | 1129 289 | 1129 290 | 112C 291 | 112C 292 |
| 112C 294 | 1139 295 | 113F 296 | 1140 297 | 1148 298 |
| 114E 299 | 115D 300 | 116A 301 | 1175 302 | 117C 303 |
| 117F 305 | 118B 306 | 11A6 307 | 1182 308 | 11B5 309 |
| 11B5 310 | 11BF 311 | 11C6 312 | 11CC 313 | 11CD 314 |
| 11D1 316 | 11DD 317 | 11E0 318 | 11EC 319 | 11F3 320 |
| 11F4 321 | 11F8 323 | 1200 325 | 1203 326 | 1209 327 |
| 120F 328 | 121A 329 | 1221 330 | 1224 332 | 1227 333 |
| 1227 334 | 122C 335 | 122C 336 | 1239 337 | 123D 338 |
| 123E 339 | 123E 341 | 124D 342 | 1252 343 | 1259 344 |
| 125F 345 | 1260 346 | 1260 348 | 1265 349 | 1271 350 |
| 128A 351 | 128D 352 | 1290 353 | 1293 354 | 1293 355 |
| 1293 356 | 1299 357 | 129F 358 | 12A5 359 | 12A8 360 |
| 12B3 362 | 12B9 363 | 12C7 364 | 12D6 365 | 12D7 366 |
| 12DE 367 | 12DE 368 | 12DF 369 | 135D 370 | 1363 372 |
| 1376 373 | 12DF 374 | 12EA 375 | 12EF 376 | 12F2 377 |
| 12F5 378 | 12FC 379 | 1302 380 | 1308 381 | 1310 382 |
| 1313 383 | 131A 385 | 1322 386 | 1325 387 | 132B 388 |

COPYRIGHT ©1982  
DIGITAL RESEARCH  
P. O. BOX 579  
PACIFIC GROVE, CA 93950

SER. # \_\_\_\_\_

|          |          |          |          |          |
|----------|----------|----------|----------|----------|
| 132B 389 | 1332 391 | 1338 392 | 1341 393 | 1347 394 |
| 1347 395 | 134D 396 | 1356 397 | 135C 398 | 1377 399 |
| 137B 401 | 1383 402 | 1384 403 | 138B 404 | 138C 405 |
| 1390 407 | 13A4 408 | 13A4 409 | 13A8 411 | 13B3 412 |
| 13B9 413 | 13BD 414 | 13BD 415 | 13BD 417 | 13C0 418 |
| 13C6 419 | 13D1 420 | 13DF 421 | 13EE 422 | 13FB 423 |
| 1403 424 | 1406 425 | 140E 426 | 1411 427 | 1419 429 |
| 1421 430 | 1424 432 | 1434 433 | 1437 434 | 1437 435 |
| 1437 436 | 143F 437 | 1442 438 | 1449 439 | 144F 440 |
| 1450 441 | 1454 443 | 145C 444 | 1461 445 | 1468 446 |
| 1470 447 | 1474 448 | 1474 449 | 147A 451 | 1480 452 |
| 1485 453 | 1491 454 | 149F 455 | 14A9 456 | 1485 457 |
| 14C5 459 | 14CA 460 | 14D3 461 | 14D6 462 | 14DB 463 |
| 14DE 464 | 14DF 465 | 14E5 467 | 14EC 468 | 14ED 469 |
| 14F5 470 | 14FA 471 | 14FF 472 | 1506 473 | 150E 474 |
| 1513 475 | 1514 476 | 1514 477 | 151C 478 | 151D 479 |
| 151D 480 | 1525 481 | 1526 482 | 1529 483 | 152A 484 |
| 152A 485 | 1539 486 | 1549 487 | 154C 488 | 154C 489 |
| 154C 490 | 1555 492 | 155C 493 | 155F 494 | 1569 496 |
| 1575 498 | 1585 499 | 1588 500 | 1588 501 | 158D 502 |
| 158D 503 | 15A1 504 | 15A1 505 | 15A8 506 | 15B0 507 |
| 15B7 509 | 15BC 510 | 15CC 512 | 15D8 513 | 15E1 514 |
| 15E4 515 | 15E7 516 | 15EC 517 | 15EF 518 | 15F4 519 |
| 15F9 520 | 15FE 521 | 15FE 522 | 160F 523 | 161A 524 |
| 162E 525 | 162E 526 | 162E 527 | 1635 528 | 1642 529 |
| 164C 530 | 164D 531 | 17A1 537 | 17A1 538 | 17B3 539 |
| 17B8 540 | 17B9 541 | 17B9 543 | 17C7 544 | 17D7 546 |
| 17E0 547 | 17E6 548 | 17E9 549 | 17E9 550 | 17F0 551 |
| 17F3 552 | 1653 553 | 1658 554 | 165D 555 | 1662 556 |
| 1665 557 | 166D 558 | 1675 560 | 1684 561 | 1690 562 |
| 1698 563 | 169B 564 | 169E 565 | 16A5 566 | 16AA 567 |
| 16B1 568 | 16BA 569 | 16BD 570 | 16C0 571 | 16C3 572 |
| 16C6 573 | 16CE 575 | 16D6 576 | 16D9 577 | 16EC 578 |
| 16EF 579 | 16F5 580 | 16F8 581 | 16F8 582 | 1700 584 |
| 1705 585 | 170A 586 | 170D 587 | 1714 588 | 171D 589 |
| 1720 590 | 1723 591 | 1726 592 | 1729 593 | 1729 594 |
| 1731 596 | 1740 597 | 1745 598 | 1748 599 | 174F 600 |
| 1758 601 | 175B 602 | 175E 603 | 1761 604 | 1764 605 |
| 1776 606 | 1776 607 | 1779 608 | 1782 609 | 1785 610 |
| 178C 611 | 178F 612 | 178F 613 | 1793 614 | 1799 615 |
| 17A1 616 | 17F3 617 | 17F3 619 | 17FA 621 | 1800 622 |
| 1803 623 | 180D 624 | 1810 625 | 1813 626 | 1813 627 |
| 181D 629 | 1822 630 | 182A 631 | 1836 632 | 1839 633 |
| 1845 634 | 1851 635 | 1852 636 | 1852 637 | 1858 638 |
| 1859 639 | 1859 640 | 1861 642 | 1864 643 | 186F 644 |
| 1872 645 | 1877 646 | 1877 647 | 188B 648 | 188B 649 |
| 188B 650 | 189A 651 | 18A1 653 | 18A6 654 | 18A9 655 |
| 18A9 656 | 18B2 657 | 18B9 659 | 18BE 660 | 18D5 662 |
| 18D8 663 | 18DB 664 | 18DE 665 | 18E1 666 | 18EA 667 |
| 18EA 668 | 1901 669 | 1907 670 | 1907 671 | 190E 672 |
| 1915 674 | 191D 675 | 1926 676 | 192E 678 | 1936 679 |
| 193E 680 | 1943 681 | 1943 682 | 1943 683 | 194B 685 |
| 1952 686 | 1958 687 | 195D 688 | 1963 689 | 1966 690 |
| 1966 691 | 196F 692 | 1974 693 | 197A 694 | 1982 695 |
| 1985 696 | 198D 698 | 1995 699 | 1998 700 | 199E 701 |
| 199E 702 | 19A8 703 | 19AE 704 | 19B4 705 | 19BA 706 |
| 19BB 707 | 19BB 708 | 19C1 709 | 19C2 710 | 19C2 711 |
| 19CF 712 | 19CF 713 | 19CF 714 | 19D5 715 | 19D6 716 |
| 19D6 717 | 19DD 719 | 19E4 720 | 19E7 721 | 19E7 722 |
| 19EA 723 | 19EA 724 | 19EA 726 | 19F0 727 | 19F8 729 |
| 19FF 730 | 1A05 731 | 1A0B 732 | 1A0F 733 | 1A12 735 |
| 1A18 736 | 1A1E 737 | 1A24 738 | 1A24 739 | 1A29 740 |
| 1A30 741 | 1A5C 742 | 1A62 743 | 1A65 744 | 1A78 745 |
| 1A7F 747 | 1A84 748 | 1A92 749 | 1A95 751 | 1A9C 752 |
| 1AA3 753 | 1AA9 754 | 1AA9 755 | 1AAC 756 | 1AB4 758 |
| 1ABA 759 | 1AC1 760 | 1AC4 762 | 1ACB 763 | 1AD2 764 |

COPYRIGHT ©1982  
DIGITAL RESEARCH  
P. O. BOX 579  
PACIFIC GROVE, CA 93950  
SER. # \_\_\_\_\_

|           |           |           |           |           |
|-----------|-----------|-----------|-----------|-----------|
| 1AD2 765  | 1AD3 766  | 1AD3 767  | 1ADA 768  | 1ADB 769  |
| 1ADB 770  | 1AE2 771  | 1AE3 772  | 1AE3 773  | 1AEA 774  |
| 1AEB 775  | 1AEB 776  | 1AF2 777  | 1AF3 778  | 1AF3 779  |
| 1AFA 780  | 1B04 781  | 1B07 782  | 1B08 783  | 1B08 784  |
| 1B0F 785  | 1B10 786  | 1B14 788  | 1B1C 789  | 1B28 790  |
| 1B2B 791  | 1B32 793  | 1B42 794  | 1B45 795  | 1B50 796  |
| 1B53 797  | 1B53 798  | 1B59 799  | 1B65 800  | 1B68 801  |
| 1B6F 803  | 1B7E 804  | 1B81 805  | 1B81 806  | 1B84 807  |
| 1B85 808  | 1B85 809  | 1B8A 810  | 1B8B 811  | 1B8B 812  |
| 1B90 813  | 1B91 814  | 1B91 815  | 1B94 816  | 1B97 817  |
| 1B98 818  | 1B9E 820  | 1BAB 821  | 1BAE 822  | 1BB1 823  |
| 1BB2 824  | 1BB2 825  | 1BBA 827  | 1BC0 828  | 1BCC 829  |
| 1BD5 830  | 1BE1 831  | 1BE4 833  | 1BEA 834  | 1BFC 835  |
| 1C05 836  | 1C10 837  | 1C10 838  | 1C11 839  | 1C51 841  |
| 1C55 843  | 1C5C 844  | 1C64 845  | 1C68 846  | 1C11 847  |
| 1C11 848  | 1C1D 849  | 1C20 850  | 1C2F 852  | 1C32 853  |
| 1C33 854  | 1C33 855  | 1C3E 856  | 1C41 857  | 1C49 859  |
| 1C4C 860  | 1C4D 861  | 1C4D 862  | 1C50 863  | 1C68 864  |
| 1C68 866  | 1C68 867  | 1C74 869  | 1C77 870  | 1C78 871  |
| 1C78 872  | 1C7B 873  | 1C8A 874  | 1C92 876  | 1C95 877  |
| 1C96 878  | 1C96 879  | 1C99 880  | 1C9A 881  | 1C9A 882  |
| 1C9D 883  | 1CA4 884  | 1CB6 885  | 1CBC 886  | 1CBF 887  |
| 1CC2 888  | 1CC3 889  | 1CC3 890  | 1CC8 891  | 1CCE 892  |
| 1CD4 893  | 1CD7 894  | 1CDE 895  | 1CE1 896  | 1CE8 897  |
| 1CEB 898  | 1CEE 899  | 1CFA 901  | 1CFF 902  | 1D02 903  |
| 1D02 904  | 1D03 905  | 1D03 906  | 1D06 907  | 1D09 908  |
| 1D0A 909  | 1D0A 910  | 1D0D 911  | 1D14 912  | 1D1F 913  |
| 1D26 914  | 1D29 915  | 1D2C 916  | 1D2D 917  | 1D2D 918  |
| 1D3A 919  | 1D3D 920  | 1D48 921  | 1D4B 922  | 1D53 923  |
| 1D56 924  | 1D57 925  | 1D57 926  | 1D78 927  | 1D78 928  |
| 1DA6 929  | 1DA6 930  | 1DB3 931  | 1DBF 932  | 1DC2 933  |
| 1D78 934  | 1D7F 935  | 1D87 937  | 1D8C 938  | 1D8F 939  |
| 1D94 940  | 1D94 941  | 1D9C 942  | 1D9F 943  | 1DA2 944  |
| 1DA5 945  | 1DC3 946  | 1DC9 949  | 1DCF 950  | 1DD4 951  |
| 1DEA 952  | 1DF4 953  | 1DFA 954  | 1E27 955  | 1E2B 956  |
| 1E32 957  | 1E35 958  | 1E38 959  | 1E3F 961  | 1E46 962  |
| 1E49 963  | 1E49 964  | 1E4D 965  | 1E4D 966  | 1E4D 967  |
| 1E52 968  | 1E55 969  | 1E5B 970  | 1E5C 971  | 1E5C 972  |
| 1E69 973  | 1E6C 974  | 1E6D 975  | 1E73 979  | 1E7E 980  |
| 1E85 982  | 1E8A 983  | 1E8D 984  | 1E8D 985  | 1EA6 986  |
| 1EB7 987  | 1EC1 988  | 1ECF 989  | 1ED3 990  | 1ED3 991  |
| 1ED3 992  | 1EE0 993  | 1EE1 994  | 1EE1 997  | 1EE4 998  |
| 1EE9 999  | 1EF1 1001 | 1EF7 1002 | 1EFD 1003 | 1F00 1004 |
| 1F06 1005 | 1F17 1007 | 1F1F 1008 | 1F22 1009 | 1F25 1010 |
| 1F2C 1011 | 1F3E 1012 | 1F46 1014 | 1F49 1015 | 1F50 1016 |
| 1F57 1017 | 1F5A 1018 | 1F5A 1019 | 1F69 1020 | 1F76 1021 |
| 1F77 1022 | 1F77 1023 | 1F7F 1024 | 1F80 1025 | 1F80 1026 |
| 1F86 1027 | 1F87 1028 | 1F87 1029 | 1F8D 1030 | 1F8E 1031 |
| 1F8E 1033 | 1F91 1034 | 1F94 1035 | 1F97 1036 | 1F9D 1037 |
| 1FA2 1038 | 1FA5 1039 | 1FA8 1040 | 1FAE 1041 | 1FCF 1042 |
| 1FD5 1043 | 1FD8 1044 | 1FD9 1045 | 1FD9 1047 | 1FE7 1048 |
| 1FEE 1049 | 1FF1 1050 | 1FF6 1051 | 1FFD 1052 | 1FFE 1053 |
| 1FFE 1054 | 2003 1055 | 2009 1056 | 200A 1057 | 200A 1058 |
| 200D 1059 | 2014 1060 | 2020 1061 | 2026 1062 | 2029 1063 |
| 202C 1064 | 202D 1065 | 202D 1066 | 2032 1067 | 2035 1068 |
| 203A 1069 | 203D 1070 | 203E 1071 | 203E 1072 | 205A 1073 |
| 205D 1074 | 205E 1075 | 205E 1077 | 2063 1078 | 206D 1079 |
| 2077 1080 | 2081 1081 | 2082 1082 | 2082 1083 | 2087 1084 |
| 208D 1085 | 2090 1086 | 2091 1087 | 2095 1089 | 20BB 1090 |
| 2088 1091 | 208C 1093 | 20C7 1094 | 20C7 1095 | 20CA 1096 |
| 20D1 1097 | 20D9 1098 | 20E3 1099 | 20E6 1100 | 20EE 1101 |
| 20F1 1102 | 20F9 1103 | 20FC 1104 | 20FF 1105 | 2102 1106 |
| 2102 1107 | 2102 1108 | 2111 1109 | 2111 1110 | 2111 1111 |
| 2117 1112 | 211E 1113 | 2136 1114 | 2139 1115 | 213C 1116 |
| 213D 1117 | 213D 1118 | 2149 1120 | 214E 1121 | 215C 1122 |
| 215F 1124 | 2164 1125 | 2170 1126 | 2170 1127 | 02BE 1128 |

COPYRIGHT ©1982  
DIGITAL RESEARCH  
P. O. BOX 579  
PACIFIC GROVE, CA 93950  
SER. # \_\_\_\_\_

|           |           |           |           |           |
|-----------|-----------|-----------|-----------|-----------|
| 02CA 1129 | 02D3 1130 | 02D8 1131 | 02EF 1132 | 02FC 1133 |
| 0309 1135 | 030F 1136 | 0312 1137 | 0312 1138 | 0321 1139 |
| 032E 1140 | 0338 1141 | 0342 1142 | 034B 1143 | 0352 1144 |
| 035A 1145 | 0362 1147 | 0368 1148 | 036B 1149 | 036E 1150 |
| 0373 1151 | 0373 1152 | 037D 1153 | 0380 1154 | 0387 1156 |
| 038D 1157 | 0393 1158 | 0398 1159 | 03A1 1160 | 03B1 1161 |
| 03B6 1162 | 03B6 1163 | 03B9 1164 | 03BE 1165 | 03D4 1166 |
| 03E1 1167 | 03E7 1168 | 03EA 1169 | 03EF 1170 | 03F5 1171 |
| 03FB 1172 | 0400 1173 | 0403 1174 | 040B 1175 | 040E 1176 |
| 0416 1177 | 0419 1178 | 0421 1179 | 0424 1180 | 042C 1181 |
| 0432 1182 | 0435 1183 | 043A 1184 | 0440 1185 | 0448 1186 |
| 044E 1187 | 0455 1188 | 045B 1189 | 0473 1190 | 047C 1191 |
| 0483 1192 | 048F 1194 | 0497 1195 | 049D 1196 | 049D 1197 |
| 04A0 1198 | 04A8 1199 | 04AD 1200 | 04AD 1201 | 04B2 1202 |
| 04B5 1203 | 04BA 1204 | 04C0 1205 | 04C9 1207 | 04CC 1208 |
| 04CF 1209 | 04D2 1210 | 04DB 1212 | 04DE 1213 | 04E3 1214 |
| 04E9 1216 | 04EF 1217 | 04F5 1218 | 04FB 1219 | 04FE 1221 |
| 0507 1222 | 0513 1223 | 0518 1224 | 0518 1225 | 051B 1226 |
| 051E 1227 | 0526 1229 | 0543 1231 | 0548 1232 | 054E 1233 |
| 0552 1234 | 0560 1235 | 0566 1236 | 0569 1237 | 0569 1238 |
| 0570 1239 | 0578 1240 | 059C 1242 | 05A4 1244 | 05A7 1245 |
| 05AA 1246 | 05AD 1248 | 05B0 1249 | 05B3 1250 | 05BB 1252 |
| 05BE 1253 | 05C1 1254 | 05C4 1255 | 05CE 1256 | 05D1 1257 |
| 05D4 1258 | 05D4 1259 | 05D7 1260 | 05DF 1262 | 05E2 1263 |
| 05EF 1264 | 05F4 1265 | 05F7 1266 | 0601 1268 | 0606 1269 |
| 060B 1270 | 060E 1271 | 0613 1272 | 0620 1273 | 0626 1274 |
| 062C 1275 | 0636 1276 | 0639 1277 | 063C 1278 | 063F 1280 |
| 064D 1281 | 0650 1282 | 0653 1283 | 0656 1284 | 0656 1285 |
| 065B 1286 | 065E 1287 | 0666 1289 | 0669 1290 | 066C 1291 |
| 067B 1292 | 0680 1293 | 0683 1294 | 06A1 1296 | 06A6 1297 |
| 06A9 1298 | 06AC 1300 | 06B7 1302 | 06BC 1303 | 06C5 1304 |
| 06C5 1305 | 06D9 1306 | 06DF 1308 | 06E7 1310 | 06EF 1311 |
| 06F6 1312 | 06FE 1313 | 070C 1314 | 070C 1315 | 070F 1316 |
| 070F 1317 | 070F 1318 | 0717 1319 | 071A 1320 | 0722 1321 |
| 072F 1322 | 0734 1323 | 0739 1324 | 073C 1325 | 073F 1326 |
| 0747 1328 | 074A 1329 | 074F 1330 | 074F 1331 | 075A 1332 |
| 075D 1333 | 0760 1334 | 0769 1336 | 076F 1337 | 0772 1338 |
| 0775 1339 | 077D 1342 | 0780 1343 | 078D 1344 | 0792 1345 |
| 0799 1347 | 07A0 1348 | 07AC 1349 | 07B6 1350 | 07B9 1351 |
| 07BF 1352 | 07C4 1353 | 07C4 1354 | 07CF 1355 | 07D2 1356 |
| 07D5 1357 | 07DA 1358 | 07E0 1359 | 07E3 1360 | 07EC 1362 |
| 07F2 1363 | 07FE 1365 | 0807 1366 | 080D 1367 | 080D 1368 |
| 0810 1369 | 0813 1372 | 0816 1373 | 081E 1375 | 0821 1376 |
| 0826 1377 | 0826 1378 | 082E 1380 | 0831 1381 | 0834 1382 |
| 0837 1383 | 083E 1385 | 0841 1386 | 0849 1388 | 084E 1389 |
| 0851 1390 | 0851 1391 | 0854 1392 | 085C 1394 | 085F 1395 |
| 0862 1396 | 0865 1397 | 086D 1398 | 0874 1399 | 0874 1400 |
| 087B 1401 | 0880 1402 | 0888 1404 | 088F 1405 | 0895 1406 |
| 089B 1407 | 089E 1408 | 08A1 1409 | 08A9 1411 | 08AC 1412 |
| 08AF 1413 | 08B2 1414 | 08BA 1416 | 08BD 1417 | 08C0 1418 |
| 08C3 1419 | 08CB 1421 | 08CE 1422 | 08D1 1423 | 08D4 1424 |
| 08DC 1425 | 08E2 1426 | 08EA 1428 | 08F1 1430 | 08F6 1431 |
| 08F9 1432 | 08FC 1433 | 08FF 1434 | 0906 1435 | 0909 1436 |
| 090C 1437 | 090F 1438 | 0912 1439 | 091A 1440 | 0920 1441 |
| 0928 1442 | 0936 1443 | 093E 1445 | 0945 1447 | 0953 1448 |
| 0958 1449 | 0960 1450 | 0963 1451 | 0966 1452 | 0975 1453 |
| 097D 1454 | 0982 1455 | 0985 1456 | 098D 1458 | 09A5 1460 |
| 09A8 1461 | 09AB 1462 | 09AE 1463 | 09AE 1464 | 09B1 1465 |
| 09C4 1467 | 09CC 1469 | 09D1 1470 | 09D7 1471 | 09DD 1472 |
| 09E0 1473 | 09E7 1474 | 09EA 1475 | 09F1 1476 | 09F4 1477 |
| 09F7 1478 | 09FC 1479 | 09FF 1480 | 0A02 1481 | 0A0A 1483 |
| 0A0D 1484 | 0A14 1485 | 0A17 1486 | 0A1A 1487 | 0A1D 1488 |
| 0A25 1491 | 0A28 1492 | 0A2B 1493 | 0A31 1494 | 0A34 1495 |
| 0A3B 1496 | 0A3E 1497 | 0A45 1498 | 0A53 1499 | 0A60 1500 |
| 0A63 1501 | 0A66 1502 | 0A6C 1503 | 0A7B 1504 | 0A7E 1505 |
| 0A8E 1506 | 0A93 1507 | 0A96 1508 | 0A9E 1509 | 0AA1 1510 |

COPYRIGHT ©1982  
DIGITAL RESEARCH  
P.O. BOX 579  
PACIFIC GROVE, CA 93950  
SER. # \_\_\_\_\_

|           |           |           |           |           |
|-----------|-----------|-----------|-----------|-----------|
| 0AA4 1511 | 0ABC 1513 | 0AC1 1514 | 0ACD 1515 | 0AD0 1516 |
| 0AEB 1517 | 0AEE 1518 | 0AF4 1519 | 0AF9 1520 | 0B03 1521 |
| 0B06 1522 | 0B0E 1524 | 0B11 1525 | 0B18 1526 | 0B25 1527 |
| 0B2C 1528 | 0B2F 1529 | 0B32 1530 | 0B35 1531 | 0B38 1532 |
| 0B3D 1533 | 0B43 1534 | 0B46 1535 | 0B49 1536 | 0B4E 1537 |
| 0B5A 1538 | 0B5D 1539 | 0B60 1540 | 0B63 1541 | 0B66 1542 |
| 0B6E 1544 | 0B71 1545 | 0B74 1546 | 0B7B 1547 | 0B7E 1548 |
| 0B8F 1549 | 0B99 1550 | 0BA6 1551 | 0BAA 1552 | 0BAD 1553 |
| 0BB0 1554 | 0BB3 1555 | 0BB6 1556 | 0BBE 1557 | 0BC4 1558 |
| 0BCC 1560 | 0BD5 1561 | 0BDA 1562 | 0BE1 1564 | 0BE6 1565 |
| 0BEC 1566 | 0BF4 1567 | 0BFA 1568 | 0BFD 1571 | 0C08 1572 |
| 0C0E 1574 | 0C13 1575 | 0C1F 1576 | 0C31 1577 | 0C37 1578 |
| 0C3F 1579 | 0C42 1580 | 0C42 1581 | 0C45 1582 | 0C57 1583 |
| 0C62 1584 | 0C6F 1585 | 0C72 1586 | 0C72 1587 | 0C75 1588 |
| 0C7D 1590 | 0C84 1592 | 0C8C 1593 | 0C8F 1594 | 0C8F 1595 |
| 0C96 1596 | 0C99 1597 | 0C9C 1598 | 0C9F 1599 | 0CA7 1600 |
| 0CAA 1601 | 0CAD 1602 | 0CB0 1603 | 0CB0 1604 | 0CB3 1605 |

COPYRIGHT © 1982  
 DIGITAL RESEARCH  
 P. O. BOX 579  
 PACIFIC GROVE, CA 93950  
 SER. # \_\_\_\_\_